1995

Simultaneous part family and machine cell formations in cellular manufacturing systems: An analytical and algorithmic approach.

A. Alper. Ozdemir
University of Windsor

National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

SIMULTANEOUS PART FAMILY AND MACHINE CELL FORMATIONS IN CELLULAR MANUFACTURING SYSTEMS: AN ANALYTICAL AND ALGORITHMIC APPROACH

By

A. Alper Ozdemir

A Thesis Submitted to the
Faculty of Graduate Studies and Research
Through the Department of Industrial
and Manufacturing Systems Engineering
in Partial Fulfilment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

1995

ISBN 0-612-01474-6

Name Alper OZDEMIR

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Engineering — Industrial

SUBJECT TERM

0546

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984

Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

To My Family

Nurten, Ahmet and Vural Ozdemir

I hereby declare that I am the sole author of this thesis. I authorize the University of Windsor to lend this thesis to other institutions or individuals for the purpose of scholarly research.

A. Alper Ozdemir

I further authorize the University of Windsor to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

A. Alper Ozdemir

The University of Windsor requires the signatures of all persons using or photocopying this thesis. Please sign below, and give an address and the date.

ABSTRACT

Increased level of global competition and rapid changes in consumers' tastes have forced the companies to reorganize their manufacturing operations, in a way that they can produce large variety of products, in small quantities, at high quality and at a low competitive price. Cellular Manufacturing Systems have been identified as the key in achieving these objectives.

In this research, mathematical programming models are developed for single and multi period planning horizons, respectively. The objective of the models is to minimize the total cost function associated with the design of cellular manufacturing systems, while minimizing exceptional part types and obtaining a balanced machining capacity distribution for single and multi period planning horizon. The mathematical models developed for single period planning horizon consider the economic trade-offs among intercellular movement, machine duplication and subcontracting, whereas the mathematical models developed for multi period planning horizon consider the economic cost trade-offs among intercellular movement, machine duplication, subcontracting and machine cell reconfiguration and recognize the fluctuations in part demand in future production periods. In order to reduce the computational time for large size problems, a heuristic technique based on the mathematical models is developed by using the Genetic Algorithms.

The mathematical models are tested by using different numerical examples and then the results are analyzed. Then, applicability of a different type of multi period planning strategy is tested and its results are compared with the multi period mathematical models in terms of arising system costs and machine cell utilizations. Finally, efficiency and applicability of genetic algorithms are tested by using a large size example and the results are compared with the mathematical models in terms of the arising cost functions, machine cell utilizations and computational time.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor Dr. S.M. Taboun, for his advice, criticism, moral and financial support during the course of this research.

I would like to thank to my committee members, Dr. Y. Aneja and Dr. M. Wang, for their advice over the past year.

I would like to thank to Mr. Tom Williams for helping me in computer related problems. I would like to extend my thanks to Ms. Jacquie Mummery for her true friendship and support since the first day I have joined the department in September, 1993.

I would like to thank to Fusun and Ahmet Alpas for helping me during my stay in windsor for the past 1.5 years. They made me feel like a part of their family.

I would like to thank to my friends Abdullah Comlekci and Tolga Unal for all the mails they have sent through the internet. They have a great share in the completion of this thesis.

Finally, I would like to thank to all my graduate friends, especially Abi M. Philipose and Michael R. Johnson, for their moral support during the preparation of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. General Overview

After years of steady economic growth and success in local protected markets, manufacturing companies around the world, particularly the ones in North America, are facing increasing level of global competition. The consumer's sense of value has been gaining variety throughout the years. Once satisfied with the low quality and low variety of products, consumer's taste is now requiring more customization to individual likes and dislikes. As a result of these changes, the manufacturing companies, which were ignorant of customer's needs, are being forced to design and reorganize their manufacturing operations, such that they can produce the custom-made products in a short period of time with high quality and variety and at a reasonable competitive price.

In order to respond to these challenges, the managers have attempted to implement the steps necessary to achieve the factory of the future which can easily compete in the future's global economy. Cellular Manufacturing Systems (CMS) have been viewed as the first step in achieving these objectives (Black, 1991). CMS are the manufacturing cells formed of machine tools, generally dedicated to the production of

a part family. They are the direct application of Group Technology (GT) philosophy. The basic idea of GT is to improve manufacturing efficiency by identifying parts and grouping them into part families and arranging the machines into groups or cells to take advantage of the similarities in manufacturing, processing and design (Sankaran and Kasilingam, 1990).

Reorganization of the functional jobshops into manufacturing cells, as illustrated in Figure 1.1, results in large amount of benefits (Wemmerlow and Hyer, 1989). These benefits can be stated as follows :

- Reductions in Throughput Time by 45.6 % .
- Reductions in Work In Process Inventory by 41.4 % .
- Reductions in Materials Handling by 39.3 % .
- Reductions in Set up Time by 32 % .
- Reductions in Labor Costs by 26.2 % .
- Improvement in Part Quality by 29.6 % .

A fundamental problem associated with CMS is to determine independent part families and machine cells, such that the machine cells do not interact with each other, i.e. the system does not contain parts moving between cells (called exceptional parts). Such independent cells simplify the scheduling operations which leads to improved shop floor control and simplified material flow. However, decomposing the manufacturing systems into independent cells is a hard goal to attain because of the existence of the

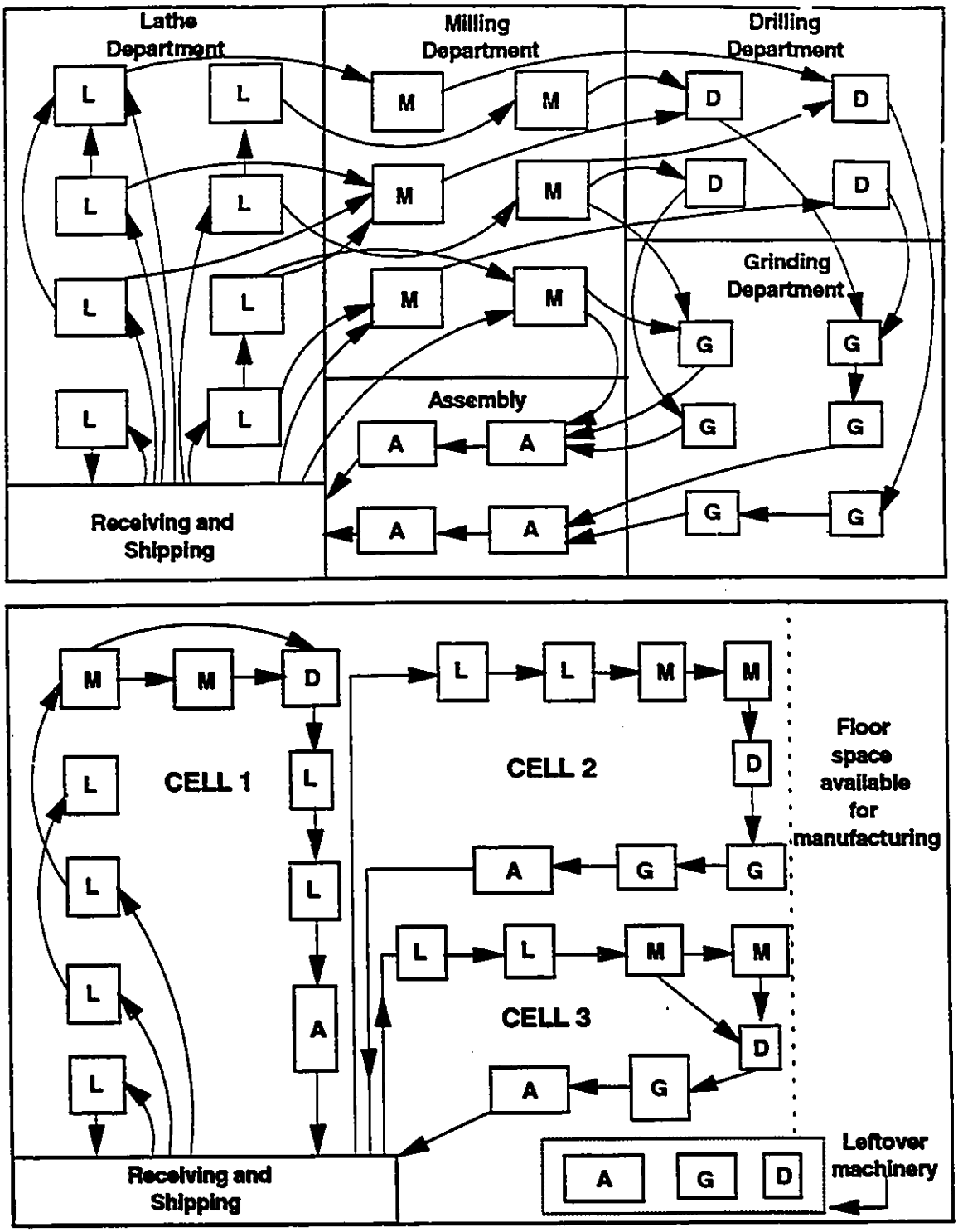

Figure 1.1. Conversion of Job Shop Systems into Manufacturing Cells.

bottleneck elements or bottleneck machines. A bottleneck machine is the one which is needed in more than one cell, whereas a bottleneck element is the one which needs machining in more than one cell. There are several approaches dealing with bottleneck elements in the literature, which can be stated as follows :

- Redesign the part such that it does not require the bottleneck machine.
- Try to allocate the operation of the part, which requires the bottleneck machine within the cell.
- Duplicate the bottleneck machine.
- Subcontract the manufacture of the part.

The decision to duplicate the machines incurs additional investment costs, as well as idle time cost unless the new duplicated machine can be fully utilized. The idle time cost on the machine represents the loss of interest on the capital invested. This alternative must be evaluated by taking into account both financial, such as machine investment and idle time costs, and non-financial considerations, such as simplified material flow and job floor control. The decision to subcontract a part, also involves financial and non-financial considerations. Financial considerations involve the evaluation of the difference between the fixed and variable costs of manufacturing and the purchase cost of buying the part from outside vendor. Objectives of the company should be incorporated into these financial analyses, for example, if the objective is to produce high quality products and maintain Just-In-Time (JIT) delivery, and if the

supplier is not able to satisfy these objectives then the financial analyst should assign higher subcontracting costs to those particular parts. Also the subcontracting costs should reflect the effect of simplified production flow by subcontracting. Non-financial considerations may involve the loss of employee morale and motivation, because large-scale subcontracting can have much the same impact on employees as plant closing or relocation of plants (Kumar and Vannelli, 1987).

1.2. Objectives of the Research

The objectives of the research are as follows:

i) To develop mathematical models for simultaneous part family and machine cell formation problems under single period planning horizon and evaluate the economic trade-offs among intercellular movement, subcontracting and machine duplication.

ii) To develop mathematical models for simultaneous part family and machine cell formation problems under multi period planning horizon and evaluate the effect of different multi period planning scenario on resulting cost functions and machine cell utilizations.

iii) To develop a heuristic technique to simultaneously develop the part families and machine cells under single and multi period planning horizons and compare the

resulting part family and machine cell formations with the mathematical models in terms of resulting cost functions and machine cell utilizations.

1.3. Organization of the Research

The research in the thesis is organized as follows. A literature review on various aspects of part family and machine cell formations in cellular manufacturing systems, and motivation for the research is presented in Chapter 2. The development and explanation of the mathematical models for single and multi period planning horizon is presented in Chapter 3. In Chapter 4, the development and application of Genetic Algorithms as a heuristic technique is presented. In Chapter 5, the mathematical models for single and multi period planning horizons are tested by using examples of different sizes to observe their validity. Efficiency and accuracy of the Genetic Algorithms are tested by using large size examples. Discussions and comparisons of the mathematical models and genetic algorithms regarding resulting cost functions, computational time and machine cell utilizations, are also presented. Conclusions and recommendations for future research are presented in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

Several clustering techniques, such as algorithmic methods and mathematical models have been developed to find the solutions for the part family and machine cell formation problems. The techniques form the part families and machine cells either simultaneously or sequentially. The review of literature on several clustering techniques presented in following sections, are as follows :

- Classification and Coding Approach
- Algorithmic Approach
- Network Approach
- Mathematical Programming Approach

2.1. Classification and Coding Approach

One of the simplest classification and coding methods is the visual classification, in which parts are grouped into part families according to their similarities of their geometric shape. This method leads to very inconsistent results because of the high dependence of the selection procedure on personal judgement. There are many reasons for this such as the fact that each person will have a different knowledge of the

processing capabilities of the machines.

The second method involves the classification of the parts into part families by examination of the individual design and/or manufacturing attributes of the parts, namely, geometric shape and complexity, dimensions, type of material, shape of raw material and the required accuracy of the finished part. The classification results in a code number that uniquely identifies the parts' attributes. Three types of code are used: hierarchial (monocode), attribute (polycode) and hybrid (mixed code). In the hierarchial code, the interpretation of each character is dependent on the meaning of the previous character in the code, whereas in the attribute code, each part attribute is independent of any character value. The hybrid code is a combination of the attribute and hierarchial codes.

Several coding schemes have been developed such as CODE, OPITZ, DCLASS and MICLASS. Although these classification and coding methods have several advantages, such as efficient retrieval of process plans, efficient means of determining the manufacturing capability and producibility, they are not widely used in great amounts because of the lack of standards among the coding schemes.

2.2. Algorithmic Approach

In the literature, several algorithmic approaches have been developed for part

family and machine cell formation problems. These approaches are: Machine Component Matrix Manipulation Approach, Similarity Coefficient Approach and Heuristic Approach.

2.2.1 Machine Component Matrix Manipulation Approach

The main objective of machine component matrix manipulation approach is to divide the components into families and machines into groups, in such a way that each component can be fully processed in one group. This approach rearranges the rows and the columns of a machine component matrix X, where rows correspond to machine types and columns correspond to components. Each element of X (x_{ij}) will take the value of one, if machine type i is required for production of component j; zero otherwise. This approach is usually employed to uncover completely decomposable blocks of X, i.e. submatrices of X such that x_{ij} is equal to zero for all pairs of (i,j) that do not belong to the same submatrix. (Shafer and Rogers, 1993).

One of the earliest approaches is the Production Flow Analysis (PFA), developed by Burbidge (1971, 1989, 1992). PFA is a technique to find a complete division of all the parts into part families and a complete division of all the existing machines into associated groups. It is based on the progressive analysis of the information contained in the route cards for the components and assemblies produced in a factory by solely

concentrating on the methods of manufacturing without attempting to change them.

King (1980) developed the Rank Order Clustering (ROC) algorithm which rearranges the rows and columns of the initial machine incidence matrix in decreasing binary values to obtain a block diagonal form. However, strong dependence of the results on the initial order of machine-part matrix and existence of storage problems created by the usage of binary value used for reallocation, restricted the applicability of the algorithm. To overcome these limitations and to improve the computational efficiency, King and Nakornchai (1982) developed ROC2.

Chan and Milner (1982) developed the Direct Clustering Algorithm (DCA) to form the part families and machine groups for cellular manufacturing systems. DCA attempts to obtain a block diagonal form and groups the rows with the "left-most" positive cell to the top and the columns with the "top-most" positive cells to the left of the machine-part matrix. The algorithm is less dependent on the initial machine-part matrix than ROC, since the procedure counts the number of positive cells in each column and row rather than using intuition.

Chandrasekharan and Rajagopalan (1986) developed MODROC, an extension of the ROC algorithm. The algorithm employs a block and slice method, and a hierarchial clustering method to obtain a set of intersecting machine cells and non intersecting part families.

Kusiak and Chow (1987) developed a cluster identification algorithm to

decompose the part-machine incidence matrix into mutually separable submatrices. The algorithm was tested on several problems and the results indicated that the algorithm was computationally efficient in terms of the CPU time. The authors also developed the cost analysis algorithm, an extension of the cluster identification algorithm, to deal with the part-machine incidence matrices which contain exceptional parts. The algorithm removes the exceptional parts based on subcontracting costs.

Chu and Tsai (1990) proposed a general procedure for evaluating three machine-part grouping algorithms, namely, the rank order clustering algorithm, the direct clustering algorithm and the bond energy algorithm. Several problems are tested by using these algorithms and the results are compared by using four different criteria: total bond energy, percentage of exceptional elements, machine utilization and grouping efficiency. The conclusions of the study have been that the bond energy algorithm had the best overall performance.

Chow and Hawaleshka (1993) proposed a new machine grouping algorithm that considers a new machine unit concept to minimize the intercellular part movements. It has been demonstrated that the total number of exceptional parts generated by the (n+1) total number of machine cells is always greater than those generated by n total number of machine cells. It has been stated that the use of the same set of computational scores to further develop new machine cells is not practical since the information from previously grouped machine cells is not included in those scores.

2.2.2 Similarity Coefficient Approach

Several researchers developed techniques to form the part families and machine cells, based on similarity coefficients. These similarity measures are generally based on the sequence of operations, the processing requirements of parts, the tooling requirements of parts and availability of the tools on the machines, the production volume and the unit operation times of parts.

De Witte (1980) proposed a method to group machines into cells based on the similarity coefficient. He defined three new similarity measures based on the absolute interdependence between machines types, on the mutual interdependence between machine types and on the relative single interdependence between machine types. To calculate these similarity measures, additional information which includes the number of available machines of each type, the routings and annual demand of each component is needed.

Waghodekar and Sahu (1984) developed a heuristic to minimize the number of bottleneck parts for cell formation problems, called MACE (machine-component cell formation), based on the similarity coefficient of the product types. Three types of similarity measures have been used namely, similarity coefficient of additive type, similarity coefficient of product type based on the total number of components processed by each pair of machines and similarity coefficient based on total flow of

common components processed by a particular machine type. The procedure produced consistent results irrespective of the sequence of machines and the parts in the initial machine component matrix.

Wei and Kern (1989) developed a new machine grouping algorithm using a new commonality score, which indicated the relative value of having two machines in the same cell (i.e. expression of the similarity of the sets of parts on which the two machines work). The algorithm considered the limitations on the cell size and the number of cells. It is flexible in the sense that various evaluation criteria can easily be incorporated.

Gupta and Seifoddini (1990) developed an algorithm for the problem of machine component grouping using a production data based similarity coefficient. The similarity coefficient takes into account the production volume, routing sequence and unit operation time for each operation. Limitations on the number of machines in a cell and the number of cells to be formed are under consideration. The algorithm identifies the part families with bottleneck machines and potential cells for their duplication by using the machine cell utilization as a criterion. However, the algorithm ignores the cost of duplicating the bottleneck machines.

Taboun et al. (1991) employed computer simulation to compare and evaluate three grouping criteria used to cluster parts and machines into part families and machine cells. Three types of similarity measures are used for the grouping criteria, namely, machine similarity, process similarity and tooling similarity. Three grouping

criteria are simulated by using SIMAN and an experimental design is conducted to observe and evaluate their performance of handling stochastic changes in the part mix and production volume levels in terms of two evaluation criteria: parts produced and the makespan. The authors have concluded that grouping derived from machine similarity was superior in terms of both evaluation criteria.

Shafer and Rogers (1993) presented a survey on similarity and distance measures used in cellular manufacturing systems, and developed a new similarity measure which eliminates the biases that results from the differences in number of parts a machine processes or from the differences in the number of machines a part requires in their processes. An experiment comparing several similarity measures based on different evaluation criteria has been presented.

Seifoddini and Hsu (1994) presented a comparative study of similarity coefficients and clustering algorithms for machine cell formation. Three different similarity coefficients: the jaccard's similarity coefficient, weighted similarity coefficient, and commonality score are compared by solving various machine component grouping problems. The results indicated that weighted similarity coefficient generated better solutions based on the number of exceptional parts.

Suer and Ortega (1994) developed a machine level based similarity coefficient to form manufacturing cells. The similarity coefficient takes into account the unit processing times of parts and the number of machines required from each machine type.

However, the authors do not consider the sequence of operations and production volumes of parts.

2.2.3 Heuristic Approach

Khator and Irani (1987) proposed the Occupancy Value method for identifying clusters in a machine-component matrix created from the route card. This method groups the parts with the highest occupancy value into part groups by progressively developing block diagonalization of the incidence matrix, and uses the part-machine incidence matrix only as an input instead of using for row-column manipulations. However, the method had difficulties dealing with exceptional elements.

Kumar and Vannelli (1987) developed a method to reorganize the existing set of parts and machines into disaggregated cells by using subcontracting as a design strategy. The method consisted of two algorithms, where the first algorithm finds the fewest number of parts to be subcontracted within operational constraints and cell size limits. The second algorithm finds the number of parts to be subcontracted which results in the least subcontracting cost within the same operating constraints.

Steudel and Ballakur (1987) proposed a new similarity measure called Cell Bond Strength (CBS), which exploits similarity in processing for pair of machines. The CBS values for the machines are calculated based on part routing and production

requirements data. A two stage dynamic programming heuristic is developed based on CBS. The first stage creates the optimum "chain" of machines, in which the sum of bonds between machines in the chain is maximized. The second stage of the heuristic partitions the "chain" to form the machine cells with the objective of maximizing the sum of CBS values in each cell, by taking into account the cell size restrictions.

Logendran and West (1990), and Logendran (1990) proposed an algorithm for part-machine grouping problems with the objective of minimizing both the intracell and intercell movements. The total movement equation is represented as a weighted sum of both intercell and intracell moves. The algorithm also considers the workload per machine in a workstation and the utilization of each machine.

Kern and Wei (1991) developed a method for identifying opportunities for reducing the number of intercellular movements, after the initial cell formation has been obtained. The method analyzes each exceptional element and generates a prioritized list suggesting the most cost effective exceptional element removal sequence and actions to be taken at each step.

Okogbaa et al. (1992) developed a heuristic algorithm to form part families and machine cells with the objective of obtaining a balanced workload distribution on the identical machines. The algorithm operates in three stages. In the first stage, machines are grouped into cells. In the second stage, the algorithm rearranges the machine cell configurations in order to minimize the inter machine flow and in the third stage, each

part is assigned to the machine cell that performs the maximum number of its operations. The heuristic algorithm is compared with the traditional process layout by using a simulation model based on three performance criteria: machine utilization, queue length and flow time.

Gupta (1993) examined the design of flexible manufacturing cells under alternative routings considerations. To group the machines into cells, a similarity index was developed which is based on alternative routing sequences, capacities of the machines, production volumes and operation times of parts. The usage factor for alternative routing sequences for each product is determined by MANUPLAN. Part families and machine cells are developed by using Complete Linkage (CLINK) as the clustering technique, such that each part is assigned to the cell that performs the maximum amount of its total operations. The author does not take into account the scheduling constraints.

Heragu and Gupta (1994) proposed a heuristic for machine cell - part family identification that addresses several important design requirements such as machine capacity, safety and technological requirements, upper limit on the size of a cell and the number of cells. First, a mathematical programming model is used to determine the number of machines from each type necessary to process the parts. The objective function minimizes the operational and capital investment costs under the machining capacity constraints. Secondly, a heuristic is employed to identify the machine cells and

their corresponding part families and to assure that all of the previously defined design constraints are satisfied. The authors do not consider the effect of the unutilized capacity of the machines on the final solution.

Ganesh and Srinivasan (1994) developed a heuristic algorithm for the cell formation problem. The algorithm considers the p-median cell formation model developed by Kusiak (1987) to form the machine cells. Parts are assigned to the machine cells where it visits maximum number of machines. Finally, an iterative scheme is employed to obtain a solution with high grouping efficacy. The algorithm has been shown to be computationally efficient.

Valle et al. (1994) proposed a workload based heuristic to form the machine cells with the objective of minimizing the intercellular movements. For each part type, total accumulated processing time in all the machine cells is calculated and the part is assigned to the machine cell with the highest value. The authors have concluded that focusing on minimization of intercellular movements rather than the minimization of intracellular movements, improves efficiency and productivity of the machine cells.

2.3. Network Approach

Rajagopalan and Batra (1975) employed a graph partitioning approach to form the machine cells. The input data in the route cards have been analyzed and used to

derive a graph whose vertices and edges represent the machines and their relationships respectively. After the creation of the cells by using the graph partitioning approach the cells are formed, the parts are allocated to the cells and the number of machines of a particular type in each cell is determined.

Vohra et al. (1990) developed a non-heuristic network approach for cell formation problems with the objective of identifying cells with a minimum level of interaction. The information in the route cards have been analyzed and used to develop a graph which represents the relationships between the machines and the parts. A modified version of Gomory-Hu algorithm is implemented to partition the graph to form the cells with minimum level of intercellular interaction. The level of interaction is measured by the amount of machining times performed outside of the parent cell. The algorithm ensures that the resultant interaction among the cells is minimized.

Lee and Hwang (1991) developed a hierarchial divisive algorithm based on graph theory to form the part families and the machine cells. The objective of the algorithm is to group the parts into part families in a way that the degree of relationship is high among the parts within the part families and low among the parts of different part families.

Boe and Cheng (1991) proposed a close neighbour algorithm to identify machine groups and form part families. The algorithm overcomes the deficiencies of the clustering and data organization algorithms and array sorting methods by producing

a block diagonal structure and eliminating the need for visual identification of part families and machine cells.

Askin et al. (1991) proposed a new method called the Hamiltonian Path Heuristic (HPH) for the algorithmic rearrangement of the part-machine incidence matrix in three stages. In the first stage, the distance matrix for the machines (parts) is calculated from the part-machine incidence matrix. In the second stage, a heuristic solution is found to the Travelling Salesman Problem by use of the distance matrix. Finally in the third stage, the results of the second stage are used to find a heuristic solution to the associated Hamiltonian Path Problem.

Wu and Salvendy (1993) proposed new algorithms for cell formation problems by the use of an undirected graph, taking into account the operation sequence constraints. The first algorithm partitions the graph into cells to minimize the resultant interaction between the cells. The second algorithm simplifies the amount of computation by selecting seed nodes in partitioning the graph.

Lee and Garcia-Diaz (1993) developed a three stage procedure for identification of part families and machine cells by using a network flow approach with the objective of minimizing the intercellular movements between the cells. The first stage computes a distance parameter for measuring the machine dissimilarity. The second stage forms a capacitated network which uses the distance parameter corresponding to machines based on the per-unit flow cost of the capacitated network. The third stage develops the

network and identifies machine cells by using the method used for solving the minimum cost network flow problems.

Chen and Guerrero (1994) developed a new approach for cell formation problems based on artificial intelligence principles. The approach uses the similarity matrix to create the machine groups and then an evaluation function is applied to select a machine cell arrangement. The approach employs a graph-based representation to represent the problem and illustrate the solution strategies.

2.4. Mathematical Programming Approach

Part family and machine cell formation problems in the literature, have been formulated by using integer and mixed integer mathematical programming models with an increasing amount of attention. These models are formulated based on several objectives such as minimization of total cost function, maximization of the similarity measures, while considering different system constraints such as capacity and budget limitations. This section summarizes the models based on their objectives under two separate headings: Models for Cell Formation and Models for Minimizing System Costs.

2.4.1 Models for Cell Formation

The models discussed in this section, analyze the part family and machine cell formation problem either sequentially or simultaneously. The objective of the models include, maximizing the similarity measure within each part family and/or machine cell, minimizing the intercellular movements and the deviation between the workload assigned to the machine and the available capacity. System constraints such as, available capacity and cell size, are under consideration.

Kusiak (1987) proposed two different models based on the p-median clustering algorithm. Both of the models use the similarity coefficients based on the machine-part incidence matrix. The first model determines the contents of the part families with the objective of maximizing the similarities of the parts within the same part family. The second model extends the first model, to include alternative process plans for each part type. The models have difficulty in assigning the initial value of p. Ben-Arieh and Chang (1994) modified the p-median clustering algorithm by introducing p, the number of machine cells, into the objective function; thus eliminating the iterative nature of the algorithm, and overcoming the difficulty of assigning an initial p value.

Co and Araar (1988) proposed a three-stage procedure for the configuration of machine cells and assignment of cells to process specific sets of jobs. In the first stage, a mathematical model is used to assign the operations to machines with the objective

of minimizing the deviations between the workload assigned to the machines and the available capacity. Available machining time is the only constraint. In the second stage, the results of the first stage are converted into machine incidence matrix and used as an input for the extended Rank Order Cluster (ROC) algorithm to group the machines based on their similarities. In the third stage, a direct-search algorithm to define the composition of the manufacturing cells is applied.

Gunasingh and Laskhari (1989) proposed a sequential 0-1 integer programming approach for the design of cellular manufacturing systems. The first model forms the machine cells with an objective of maximizing the similarities between machines within the same cell under the cell size constraint. The index of "similarity" between two machines is defined in terms of the tooling requirements of the parts (processing requirement), and the tools available on the machines (machine capabilities). The second model assigns the parts to the machine cells with an objective of maximizing the compatibility of the machine cell and the part assigned to them under the part family size constraint. The index of "compatibility" between a part and the machine cell is defined by the extent to which the processing requirements of a part can be met by the machine cell. The model ignores the possibility of having multiple units of a particular machine type in the same cell.

Gunasingh and Lashkari (1989) developed two 0-1 integer programming formulations to group the machines in cellular manufacturing systems based on the

tooling requirements of the parts, tooling capacities on the machines and the processing times. The formulations assume that the part families are known apriori. The first formulation groups the machines based on the compatibility of parts with machines. The second formulation groups the machines in order to minimize the machine allocation and intercell movement costs. Limitations on the number of machines in a group and the number of machines available from each type are under consideration.

Taboun and Sharma (1991) introduced a weighted similarity index and incorporated into a mathematical programming model to form part families and machine cells. It is a mixed 0-1 non-linear model with a quadratic objective function which attempts to maximize the similarities of the part pairs assigned to cells under the capacity constraints. Three types of similarity indexes are studied which are based on processing machines, sequence of operations and tooling requirements.

Logendran (1991) observed the effect of the identification of key machines in cell formation problems and proposed a mathematical model to minimize the total moves by taking into consideration the sequence of operations exhibited by parts. The total moves contributed by all parts have been evaluated as a weighted sum of both inter and intra cell moves. A heuristic solution, including two algorithms, have been developed for the model and sensitivity of three proposed methods have been analyzed with regard to identifying the key machines and their impact on the selection of final part-machine clusters.

Dahel and Smith (1993) developed two mathematical models which simultaneously group parts and machines into predefined number of cells. The first model minimizes intercell moves subject to machine capacity and cell size constraints. The second model is formulated as a multiobjective model which is used to form cells which are both flexible and have minimum interactions. Two models are analyzed under the intercell routing flexibility criteria.

Logendran (1993) proposed a quadratic binary mathematical model for evaluating the effectiveness of simultaneous part-machine groupings in cellular manufacturing systems. The objective function is formulated as the maximization of a unified measure of effectiveness evaluated as the weighted sum of fractions representative of the (negative of) total moves and the in cell utilizations based under the operational constraints associated with setting up a cellular manufacturing system such as processing time requirements of parts on machines, sequence of operations, machine capacities and non-consecutive operations scheduled to be performed on the same machine. The model can be extended to include multiple routeings for each part.

2.4.2 Models for Minimizing System Costs

The models discussed in this section form the part families and the machine cells with the objective of minimizing the system costs. These costs usually include annual

setup costs, average annual WIP costs, annual average production costs, capital machine investment costs, material handling costs and subcontracting costs. System constraints such as available machining capacity, cell size limits and budget restrictions are under consideration.

Choobineh (1988) proposed a two-stage procedure for the design of cellular manufacturing systems. In the first stage, a new similarity measure is developed to form the part families, which is an extension of Jaccard's similarity coefficient by McAuley (1972), with an inclusion of sequence of common operations. In the second stage, an integer programming model is developed to form the machine cells. The objective function includes the total of annual setup, average annual WIP cost, annual average production costs and machine investment cost. The model only considers the available machining capacity and available operating budget restrictions.

Shtub (1989) developed a generalized assignment model for the cell formation problem with the objective of minimizing the cost of assigning tasks to agents. Several examples have been solved and it has been shown that cell formation problem, with several process plans under consideration, is equivalent to the generalized assignment problem.

Sundaram (1989) developed a model for design of cellular manufacturing systems with alternate routing considerations. The objective of the model is to find the best possible routing and the machine centres for each part, at the least possible capital

investments on machining centres. Demand of the parts and available machining capacities are taken into account. The model may be extended to include intercellular material handling cost.

Askin and Chiu (1990) developed mathematical models for cell formation and machine assignment problems. The objective of the first model is to form the machine cells and allocate the operations to the machines with the objective of minimizing machine overhead, tooling costs and intercellular movement costs. A two stage formulation, based on graph teoric heuristic, was developed to simplify the solution procedure. The first stage assigns the operations of the parts to the individual machines and the second stage assigns the machines to the groups, with the objective of minimizing intercellular material handling cost.

Sankaran and Kasilingam (1990) developed two mathematical models for cell formation and part routing selection. The objective of the models is to maximize the routing flexibility of the system as well as minimize the sum of processing costs and annualized machine operation costs, while forming the cells and selecting the best routing for each part. Sankaran (1990) developed two mathematical models for cell formation. The first model is developed as a multiple objective mathematical model for cell formation. The objective function is the sum of five distinct linear cost functions which consist of capital costs of machines, machining costs of parts, tool usage costs, process costs exclusive of machining and tooling and material handling costs between

machines. Tool and machine similarity with limitations of available machining time, part movement and the number of parts assigned to each cell are under consideration in the formulation. The second model is formulated as a goal programming formulation. The objective function considers the weighted sum of the deviations from the desired aspiration levels.

Rajamani et al. (1990) proposed three mathematical models to analyze the affects of alternative process plans on the resource utilization when the part families and machine groups are formed simultaneously. The first model assigns machines to parts. The objective function is to minimize the total investment cost under the machine capacity and budget constraint. The second model assumes that part families are known apriori. It selects a process plan for each part, machine type for each operation and number of machines of each type in different cells. The objective function is to minimize the total investment on machines of different types assigned to the cells under the same constraints as specified in the second model. The third model identifies part families and machine groups simultaneously. The objective functions and the limitations are basically the same. Comparisons of the three models based on cost functions are also given.

Gunasingh and Laskhari (1991) proposed a methodology to simultaneously group machines and parts in cellular manufacturing systems based on the tooling requirements of the parts, tools available on the machines and the processing times.

Two non-linear 0-1 integer programming formulations are proposed. The first formulation forms machine-part groups on the basis of the compatibility of the parts with the machines. The second formulation groups the machines and the parts in order to seek a trade-off between the cost of duplicating the machines and the cost of intercell movements.

Jain et al. (1991) developed an 0-1 integer programming model to form the cells and provide the tools in a flexible manufacturing system. The objective of the problem is to minimize the overall system cost, defined as the sum of the annual cost of processing the parts, cost of tools and annualized cost of machines. Processing requirements of the parts, available processing times on the machines and tool lives are considered in the model.

Damodaran et al. (1992) developed a mathematical model to simultaneously form machine groups and assign operations of the parts to these machine groups. The model identified the cost trade-offs among intercell movements, refixturings and the duplication of machines, while taking into consideration the existence of multiple process plans for each part type.

Liang and Taboun (1992) developed a model to address the simultaneous part selection and assignment problems in flexible manufacturing systems with existing layouts, while dealing with exceptional parts and inter-cell movements. The objective function maximizes the total profit while minimizing the inter-cell movement costs,

machining costs and high penalty costs toward the unsatisfied due dates.

Shafer et al. (1992) developed a mathematical model to deal with exceptional parts by considering the cost trade-offs among intercellular movement costs, subcontracting costs and machine duplication costs. It was assumed that, sets of exceptional parts, bottleneck machines in the cells, annual demand of parts and annual capacities of machines are known in advance. The model assumed that the initial part family and machine cell formation, has been obtained by using one of the existing techniques, but ignored the importance of the solution technique on the exceptional elements. The model may be extended to find a simultaneous solution for the part family and machine cell formation problems, while dealing with the exceptional elements.

Rajamani et al. (1992) developed a mathematical model for cell formation problems where there are significant sequence dependent setup times and costs. The mathematical model determines the economic number of cells with the objective of minimizing the sum of total discounted cost of machines assigned to the cells and the setup costs incurred due to sequence dependence of parts in each cell. Machine capacity and precedence relationships of the parts are under consideration.

Adil et al. (1993) developed a mathematical model to form the machine cells by considering the cost trade-offs between the investment cost and the operational cost. The investment cost includes the cost of operating the cells and the machines, whereas

the operational cost includes the cost of sequence dependent setup, machine idle time, WIP inventory, part early and late finish. The model determines the economic number of cells, allocation of the parts, scheduling and sequencing of parts within the cells. The authors reported that the consideration of WIP and machine idle time increased the computational times.

Riberio and Pradin (1993) proposed a two-phase methodology to organize a job shop production system into a cellular manufacturing system. The first phase (selection/assignment) selects the machines to be kept on the shop floor and assigns parts to the machines retained by taking into account the machining capacity and the part demand. The second phase (partition/reassignment) establishes a partition of the set of parts and corresponding machine cells and reassigns some of the operations with the objective of eliminating some intercell part movements.

Atmani et al. (1994) developed a zero-one integer mathematical model for simultaneous machine cell formation and operation allocation problems in Cellular Manufacturing Systems. The model considers multiple process plans for each part type. The objective of the model is to group the set of existing machines into machine cells and allocate the operations of the parts to the grouped machines in order to minimize the sum of operation, refixturing and transportation costs. The model ignores the option of machine duplication and the effects of machine costs on the final optimal solution. Atmani et al. (1994) extends the formulation in the previous paper to include

the machine costs. To investigate the effect of machine cost on the cell design, these three different machine cost models are introduced machine variable time cost, machine variable time cost plus penalty cost on the fraction of time machine type j is not productive and machine variable time plus machine fixed cost. The models are tested with various examples to check their validity and sensitivity analysis for changes in the model parameters are included.

Liao (1994) developed a three stage procedure for designing a line-type CMS with the objective of minimizing operating and material handling costs. At the first stage, a mathematical model is developed to select the best routeing for each part type with the objective of minimizing operating costs. In the second stage, the machine cells are formed by using the results of the first stage as the input to a neural network model. Finally in the third stage, a facility layout software package is used to determine the layout of the machine cells with the objective of minimizing the material handling costs.

Liang (1994) proposed a two-stage approach to jointly solve part selection, machine loading and machine speed selection problems in Flexible Manufacturing Systems. In the first stage, a mathematical model is developed to solve the part selection and machine loading problem with the objective of maximizing the system output (e.g. production quantity, dollar value of the parts produced). Available machining capacity and magazine capacity constraints are under consideration. In the second stage, a mathematical model is employed to find the optimal cutting speed for all possible job,

tool and machine combinations with the objective of minimizing the sum of labour, capital, machining overhead and tool costs. Available machining time is the only constraint.

2.5. Motivation for the Research

The literature review indicates that several techniques and models, which form the part families and machine cells either sequentially or simultaneously, have been developed. Many of the existing mathematical models consider several cost functions such as material handling, machine investment, set up, tooling and processing costs, but generally the cost of unutilized capacity has been ignored. Even though there exist models that consider economic cost trade-offs among refixturing, intercellular movement and machine duplication; there are not many models that consider the option of subcontracting. The models that consider the option subcontracting; solve the problem sequentially and try to minimize the exceptional parts in a predefined part family and machine cell formations by considering the economic trade-offs among intercellular movement, machine duplication and subcontracting. Hence there exists a need to develop mathematical models which will simultaneously form the part families and machine cells by considering the economic trade-offs among intercellular movement, machine duplication and subcontracting.

Most of the existing mathematical models in the literature focus only on single production periods without considering the possibility of fluctuations in part demand and changes in workload assignments to the machine cells, in future production planning periods. Thus, there exists a need to develop mathematical models which will develop simultaneously the part family and machine cell formations for each period in the planning horizon by considering the fluctuations in part demand in each period. Also, the computational time to reach the optimal solution by using the mathematical models increases exponentially as the problem size gets bigger, due to the increase in the number of integer variables. Therefore, it would be beneficial to develop a heuristic technique which considers the same cost functions and operating constraints specified in the mathematical models, in order to reduce the amount of computational time for large size problems.

CHAPTER 3

MODEL DEVELOPMENT

This chapter presents mixed integer mathematical models for simultaneous part family and machine cell formation problems in cellular manufacturing systems under single and multi period planning horizons. The first model evaluates the cost trade-offs between machine duplication and intercellular movements, whereas the second model evaluates the cost trade-offs among subcontracting, machine duplication and intercellular movements under single period planning horizon. The third model evaluates the cost trade-offs among machine duplication, intercellular movement and machine relocation, whereas the fourth model evaluates the cost trade-offs among subcontracting, machine duplication, machine relocation and intercellular movements under multi period planning horizon. Accordingly, this chapter is divided into two main sections: Section 1, for models developed under the single period planning horizon and Section 2, for models developed under the multi period planning horizon. In these sections the general definition of the problems, statement of the assumptions, explanation of the variables, parameters used in the design and development of the models are included.

3.1. Mathematical Models Under Single Period Planning Horizon

3.1.1. Problem Definition

The main objective of cellular manufacturing systems, is to decompose a manufacturing system into several subsystems such that each subsystem can operate independently. However, in real life applications, existence of exceptional components usually makes this objective hard to attain. The mathematical models developed in this section, attempt to minimize the exceptional parts by considering the options of duplicating some of the machines, creating intercellular movements and subcontracting some of the parts, based on several cost values. Some of these options can be explicitly stated as follows:

- Duplicating the machine and creating unutilized capacity by incurring additional investment cost and idle time cost but saving from intercellular movement cost
- Allocating the operation of the part to another machine of the same type in another cell and creating intercellular movements by incurring intercellular movement cost but saving from additional investment cost and excess idle time cost.

- Subcontracting a part by incurring subcontracting costs and increased idle time cost but saving from machining costs, additional investment cost and intercellular movement cost.

3.1.2. Assumptions

1) Parts to be produced for the planning horizon are known and have constant demand throughout the planning horizon.
2) For each part type route cards, which give information about the sequence of operations to be performed, type of machines required and processing times on these machines are available.
3) If a part type has more than one operation on the same particular type of machine, all the operations are combined and treated as a single operation.
4) Data on available machining capacity and various costs are known.
5) There exists a unique process plan for each part type and each part type's operation can only be processed on a single machine.
6) Intercellular movements among the cells are allowed and intercellular movement cost per movement is assumed to be constant for each part type.
7) Intracellular movements are assumed to be negligible since parts with similar process requirements will be grouped into the same cell.

8) The number of machine cells to be formed is determined beforehand by the decision maker.

9) There can be multiple units of machines from each type in the system and cost of each machine is assumed to be the same for a given machine type.

10) Subcontracting is allowed, but once the specific part type is subcontracted, the whole demand of that part type must be subcontracted.

3.1.3. Nomenclature

i. Indices

i	Part Type Index	i=1,...,N
j	Cell Index	j=1,...,CN
k	Machine Type Index	k=1,...,K

ii. Parameters

AC_i	Average intercell movement cost for part type i per movement.
CIM_k	Unit capital investment cost for type k machine.
D_i	Demand of part type i over the single period planning horizon.

IDC_k Unit idle time cost for type k machine, per machine hour.

K_i Set of machines required to produce part type i.

LMT_i Last machine type required by part type i, in its operation sequence.

MC_k Machining cost, $ per machine hour on type k machine.

$UMC_{k,j}$ Unit machining capacity for type k machine in cell j over the planning horizon.

MCL_j Minimum total number of machines allowed in cell j.

MCU_j Maximum total number of machines allowed in cell j.

N_i Total number of operations required to produce part type i.

SC_i Unit Subcontracting cost of part type i.

$t_{i,k}$ Unit machining time of part type i on type k machine.

iii. Decision Variables

a. 0/1 Integer Variables

$Z_{i,k,j}$

1 If the operation of part type i which requires type k machine is performed in cell j.

0 otherwise.

$M_{i,k,j}$ { 1 If part type i is transported to another cell, after it has been processed on type k machine in cell j.

0 otherwise.

S_i { 1 If part type i is subcontracted

0 otherwise.

b. General Integer Variables

Y_{kj} Total number of type k machine(s) required in cell j.

c. General Variables

ID_{kj} Total idle time of type k machine(s) in cell j.

3.1.4. Single Period Mathematical Models

In this section two mathematical models have been developed to simultaneously create the machine cells and form the part families under the single period planning horizon. The objective of the models is to minimize the total cost function related with the design of cellular manufacturing systems.

3.1.4.1. Model I, Single Period

Model I attempts to minimize the total cost function while simultaneously grouping the machines into machine cells and forming the part families in order to obtain independent machine cells. As long as there is available capacity on the machines in the cell, the model will allocate the operations of these parts to the machines. If available capacity does not exist on one of these machines, then the model will select one of the following options with the least cost.

- If there are not any machines of the required type in other machine cells, then the model will create additional capacity in the cell by allocating additional machines of the required type to the machine cell, to satisfy the production requirements.
- If there are other machines of the required type in other cells and have enough capacity to meet the capacity requirements of the specific operation of the part, then the model will allow intercellular movement and allocate the operation of the part to the machine which is in another cell, as long as, the cost of intercellular movement is less than the total cost of buying another machine and the cost of unutilized capacity of the duplicated machine.

Given the notations and explanations of the options, a mixed integer mathematical programming model for simultaneous part family and machine cell formation under single period planning horizon, is developed as follows:

Minimize

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\left[CIM_k\ Y_{kj} + IDC_k\ ID_{kj}\right] + \sum_{i=1}^{N}\sum_{k=1}^{K_i}\sum_{j=1}^{CN} AC_i\ M_{ikj}$$

subject to the following constraints:

$$\sum_{j=1}^{CN} Z_{ikj} = 1 \qquad \forall\ i \in N,\ k \in K_i \tag{3.1}$$

$$Z_{ikj} - Z_{ik+1j} \le M_{ikj} \qquad \forall\ i \in N,\ k \in K_i,\ k \notin LMT_i\ ,\ j \in CN \tag{3.2}$$

$$\sum_{i=1}^{N} D_i\ t_{ik}\ Z_{ikj} + ID_{kj} = UMC_{kj}\ Y_{kj} \qquad \forall\ k \in K_i,\ j \in CN \tag{3.3}$$

$$MCL_j \le \sum_{k=1}^{K} Y_{kj} \le MCU_j \qquad \forall\ j \in CN \tag{3.4}$$

$$Z_{ikj}\ ,\ M_{ikj} \in (0,1) \qquad \forall\ i \in N,\ k \in K_i,\ j \in CN \tag{3.5}$$

$$integer \quad Y_{kj} \qquad \forall \; k \; e \; K \tag{3.6}$$

The objective of the model is to minimize the various cost functions related with the design of cellular manufacturing systems under single period planning horizon. There are three components considered in the objective function, which are: total capital investment cost, total idle time cost and total intercellular movement cost.

The first component in the total cost function is the total capital investment cost related with acquiring different types of machines in the system. Capital investment cost for each type of machine is found, by calculating the annualized equivalent of the investment cost over the planning horizon. The investment cost includes the initial purchase price of the machinery and the salvage value at the end of the useful life. Capital investment cost can also be considered as the cost of depreciating the machinery over the planning horizon.

The second component is the total idle time cost, which assigns a penalty cost on the unutilized capacity of the machine. The penalty for the idle time of the machine($/hr) is defined as the opportunity cost of not processing the part on the machine which is equal to the loss of interest on the investment. Additionally, assigning penalty cost on the unutilized capacity of the machine will balance the utilization of the machines in the system without creating excess idle capacity.

The third component in the total cost function is the total intercellular movement cost which assigns a fixed amount of cost to each movement among cells. Average intercellular movement cost may include several components, such as the operation and investment cost of the special material handling equipment, cost of special fixtures, cost of controlling the increased material flow between the cells and cost of training the operators for the new part type. Even though, allowing intercellular movements contradicts with the group technology philosophy, in practical situations it might be applicable, as long as, it is economically feasible.

The model developed in this section considers six different operating constraint sets related with the design of cellular manufacturing systems.

It was assumed that multiple units of the same type of machine can exist in the cells and each operation of specific part type can only be allocated on a single type of machine. Constraint set (3.1) ensures this integrality by allocating each operation of part type i, to be processed on machine type k, in one of the machine cells.

One of the objectives of the model is to minimize the intercellular movements by grouping the similar parts and machines in the same cells. Constraint set (3.2) motivates the model to minimize the intercellular movements, thus the exceptional elements, by trying to group the operations of the part types in the same cell, since each time a part makes a movement outside of the cell, the indicator variable $M_{ik,j}$ will have the value of 1 and an intercellular movement cost will be assigned to that part. In the constraint,

index (k+1) points to the next machine, after machine type k in the machine set K_i, not to the next machine in the sequence.

Each specific machine type in the system, has a limited available machining capacity and violation of this capacity is not allowed. Constraint set (3.3) ensures that in each machine cell enough machining capacity is available for each type of machine to satisfy the production requirements. For each type of machine in the cell, an equilibrium is reached among the demand for capacity, unutilized capacity of the machine and the available capacity.

Constraint set (3.4) brings a lower and an upper limit on the total number of machines in each cell. This constraint has very important consequences, since the high number of machines in the cell makes the control of the cell very difficult because of the increased intra cell material flow, which will in turn decrease the effectiveness of the system. Also having a few number of machines in the cell may not be economically feasible, as there are several costs included in management of a cell.

Constraint sets (3.5) and (3.6) ensure the integrality of the model. The model presented above will have ($K \times CN$) number of integer variables, where K is the number of machine types and CN is the number of machine cells. For each part type, it will have ($CN \times N_i$) number of $Z_{i,k,j}$ binary variables and ($CN \times (N_i-1)$) number of $M_{i,k,j}$ binary variables.

3.1.4.2. Model II, Single Period

The objective of Model II is to minimize the total cost function while simultaneously grouping the machines into machine cells and forming the part families. The total cost function is the sum of capital investment cost, idle time cost, intercellular movement cost, machining cost and subcontracting cost. Model II, in addition to considering the alternatives of machine duplication and intercellular movements as discussed in model I, also considers the option of subcontracting the whole production of some of the part types. The model considers several different strategies for part family and machine cell formation, some of which can be stated as follows:

- If there is enough capacity on the machine to process the operations of the part in the cell, then the model will allocate the operations of the parts to the machines and produce the parts within the cell as long as cost of machining the part within the cell is less than the sum of cost of subcontracting the part and cost of unutilized capacity created by subcontracting.
- If there is not enough capacity in the cell to process some of the operations of the parts, then the model will select one of the three following actions, with the minimum cost.

• Create enough capacity within the cell by acquiring additional machines from the required type. The cost of this action is the sum of total capital investment cost of the additionally acquired machines and the total cost of unutilized capacity from the additionally acquired machines.

• If there is available capacity on the other machines from the required type in other cells to process the operations of the part types, allocate these operations to these machines and create intercellular movements. The cost of this action is the sum of intercellular movement costs created by allocating some of the operations outside of the cell.

• Subcontract all the operations of the part types, which require extra capacity. The cost of this action is the sum of total subcontracting costs and total cost of unutilized capacity created by subcontracting the parts.

Given the notations used in the formulation of the model and an explanation of the possible options, a mixed integer mathematical programming model, for simultaneous part family and machine cell formation is developed as follows:

Minimize

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\left[CIM_k\ Y_{kj} + IDC_k\ ID_{kj}\right] + \sum_{i=1}^{N}\sum_{k=1}^{K_i}\sum_{j=1}^{CN} AC_i\ M_{i,k,j}$$

$$+ \sum_{i=1}^{N}\sum_{k=1}^{K_i}\sum_{j=1}^{CN} D_i\ t_{i,k}\ MC_k\ Z_{i,k,j} - \sum_{i=1}^{N} SC_i\ D_i\ S_i$$

subject to the following constraints:

$$\sum_{j=1}^{CN} Z_{i,k,j} \le 1 \qquad \forall\ i \in N,\ k \in K_i \qquad (3.7)$$

$$Z_{i,k,j} - Z_{i,k+1,j} \le M_{i,k,j} \qquad \forall\ i \in N,\ k \in K_i,\ k \notin LMT_i\ ,\ j \in CN \qquad (3.8)$$

$$\sum_{i=1}^{N} D_i\ t_{i,k}\ Z_{i,k,j} + ID_{kj} = UMC_{kj}\ Y_{kj} \qquad \forall\ k \in K_i,\ j \in CN \qquad (3.9)$$

$$\frac{1}{N_i}\left[\sum_{k=1}^{K_i}\sum_{j=1}^{CN} Z_{i,k,j}\right] + S_i = 1 \qquad \forall\ i \in N \qquad (3.10)$$

$$MCL_j \le \sum_{k=1}^{K} Y_{kj} \le MCU_j \qquad \forall\ j \in CN \qquad (3.11)$$

$$Z_{i,k,j}\ ,\ M_{i,k,j}\ ,\ S_i \in (0,1) \qquad \forall\ i \in N,\ k \in K_i,\ j \in CN \qquad (3.12)$$

$$integer \quad Y_{kj} \qquad \forall \; k \in K \tag{3.13}$$

The objective of the model is to minimize the various cost functions related with the design of the cellular manufacturing systems. There are five cost components considered in the total cost function, which are the total capital investment cost, the total idle time cost, the total intercellular movement cost, the total machining cost and the total subcontracting cost.

The first three components in Model II are exactly the same as in Model I. The additional cost components are the total cost of machining the parts in the system and the cost of subcontracting the part types. The machining cost for a specific type of machine is assumed to be the same regardless of the part type, which includes the tool usage cost, the fixture costs and the handling cost which is directly proportional to the time to load and unload the workpiece. Usually, subcontracting cost is determined by taking into account the financial considerations such as the cost of producing the part within the system, the cost of buying the part from the vendor and the non-financial considerations such as the simplified material flow within the system, quality of the finished parts, dependability of the vendor for longterm relationships.

Model II operates under seven different operating constraint sets related with the design of cellular manufacturing systems.

Constraint set (3.7) ensures that each operation of the part type may be allocated

to the machine it requires in one of the cells. If the part is not subcontracted, constraint (3.7) prevents the model to assign the same operation of the part type to different machine cells more than once. Constraint set (3.8) and (3.9) are exactly the same as in Model I, which controls the intercellular movements among the cells and the satisfaction of the capacity requirements in each cell, respectively.

It was assumed that subcontracting is allowed, but once subcontracted the whole demand of the part type must be subcontracted and there is no possibility of not satisfying the demand of the part type. Even though it might increase the utilization of the existing capacity, the option of subcontracting a portion of the demand is not taken into consideration, since that might result in unbalanced workload distribution and increased material flow. Constraint set (3.10) ensures that parts are either produced in the system or subcontracted. Constraint sets (3.11), (3.12) and (3.13) are the same as in Model I, where the last two constraint sets ensure the integrality of the model.

The model presented above will have ($K \times CN + N$) number of integer variables of which ($K \times CN$) of them will be related with the number of machines from each type in the cells and N of them will be related with the subcontracting of the parts, where K is the number of machine types and CN is the number of machine cells. For each part type, it will have ($CN \times N_i$) number of $Z_{i,k,j}$ binary variables and ($CN \times (N_i-1)$) number of $M_{i,k,j}$ binary variables, where N_i is the total number of operations required for each part type i.

3.2. Mathematical Models Under Multi Period Planning Horizon

3.2.1. Problem Definition

The mathematical model developed in this section, as contrary to the models that have been published in the literature which only focus on single production period, considers the possibility of changes in part demand in future production plans while simultaneously developing the part family formations and machine cell configurations for each period in the planning horizon.

The objective of the models is to minimize the total cost function while attempting to minimize the exceptional elements and obtaining a balanced machining capacity distribution throughout the production planning horizon by considering the options of duplicating some of the machines, creating intercellular movements, subcontracting some of the parts and changing the configuration of the machine cells based on several cost values. The first three options are exactly the same as stated in section 3.1 and they function within the planning periods. The option of changing the configuration of the machine cells forces the model to achieve an overall balanced distribution of the machining capacity by creating a link among the planning periods. This option can be explicitly stated as follows:

- If there is an excess machining capacity for a particular machine type in the next period because of the decrease in the demand, the excess machining capacity might be reduced by removing the machine(s) of the required types by incurring relocation cost because of the removal of the machine from the system and possible intercellular movement cost because of the change in configuration of the machine cells, but saving from the additional investment and idle time cost, as long as, the total cost of relocating the machines and possible intercellular movement is lower than the total investment and idle time cost.

- If there is an extra capacity requirement for a particular machine type in the next period because of the increase in demand, extra machining capacity might be increased by acquiring additional machine of the required type by incurring relocation cost(because of installing the machine to the system), additional machine investment, idle time cost, in order to satify the production requirements of the system, or the part(s) which creates the need for an extra capacity might be subcontracted by incurring subcontracting cost but saving from additional machine investment, idle time and relocation costs.

- If there are not any changes in the total capacity requirements for all the machine types in the system, the configuration of the machine cells might still change to

obtain a more balanced distribution of available machining capacity in order to reduce the number of intercellular movements, as long as, the saving obtained from reduced intercellular movements is higher than the cost of relocating the machines in the system.

3.2.2. Assumptions

1) Parts to be produced for the planning horizon are known and have constant demand for each period in the planning horizon.
2) For each part type route cards, which give information about the sequence of operations to be performed, type of machines required and processing times on these machines are available for each period in the planning horizon.
3) If a part type has more than one operation on the same machine, all the operations are combined and treated as a single operation.
4) Duration of each planning period, data on available machining capacity and various costs are known.
5) There exists a unique process plan for each part type and each part type's operation can only be processed on a single machine.
6) Intercellular movements are allowed and intercellular movement cost per movement is assumed to be constant for each part type in each planning period.

7) The number of machine cells to be formed in each period is determined beforehand by the decision maker.

8) There can be multiple units of machines from each type in each period in the system and the cost of each machine is assumed to be the same for a given machine type in each period.

9) Machines can be bought and sold at the end of each period and relocation cost, as a result of installation or removal, is assumed to be the same for all the machine types in each period.

10) Changes in part family composition and machine cell configuration are allowed at the end of each period, as long as, it is economically feasible.

11) Inventory carrying to the next periods is not allowed.

12) Subcontracting is allowed in each period, but once a specific part is subcontracted, the whole demand of that part must be subcontracted.

13) There is no production loss due to the changes in the machine cell configurations.

3.2.3. Nomenclature

i. Indices

i	Part Type Index	i=1,...,N

j	Cell Index	j=1,...,CN
k	Machine Type Index	k=1,...,K
t	Period Index	t=1,...,T

ii. Parameters

$AC_{i,t}$	Average intercell movement cost for part type i per movement in periodt.
$CIM_{k,t}$	Unit capital investment cost for type k machine in period t.
$D_{i,t}$	Demand of part type i in period t.
$IDC_{k,t}$	Unit idle time cost for type k machine, per machine hour in period t.
$K_{i,t}$	Set of machines required to produce part type i in period t.
$LMT_{i,t}$	Last machine type required by part type i, in its operation sequence in period t.
$MC_{k,t}$	Machining Cost, $ per hour on type k machine in period t.
$UMC_{k,j,t}$	Unit machining capacity for type k machine in cell j in period t.
$MCL_{j,t}$	Minimum total number of machines allowed in cell j in period t.
$MCU_{j,t}$	Maximum total number of machines allowed in cell j in period t.
$N_{i,t}$	Total number of operations required to produce part type i in period t.
$RLC_{k,j,t}$	Average unit relocation cost of type k machine in cell j if type k machine has been relocated at the end of period t.

$SC_{i,t}$ Unit subcontracting cost of part type i in period t.

$t_{i,k,t}$ Unit machining time of part type i on type k machine in period t.

iii. Decision Variables

a. 0/1 Integer Variables

$Z_{i,k,j,t}$
- 1 If the operation of part type i which requires type k machine is performed in cell j, in period t.
- 0 otherwise.

$M_{i,k,j,t}$
- 1 If part type i, is transported to another cell, after it has been processed on type k machine in cell j, in period t.
- 0 otherwise.

$S_{i,t}$
- 1 If part type i is subcontracted in period t.
- 0 otherwise.

b. General Integer Variables

$Y_{k,j,t}$ Total number of type k machine(s) required in cell j in period t.

$PC_{k,j,t}$ Positive change in the number of type k machines in cell j from period t to t+1.

$NC_{k,j,t}$ Negative change in the number of type k machines in cell j from period t to t+1.

c. **General Variables**

$ID_{k,j,t}$ Total idle time of type k machine(s) in cell j in period t.

3.2.4. Multi Period Mathematical Models

In this section, two mathematical models been developed to simultaneously create the part family formations and machine cell configurations for each period in the planning horizon.

3.2.4.1. Model III, Multi Period

Model III attempts to minimize the overall total cost function for the multi period planning horizon, by simultaneously grouping the machines into machine cells and parts into part families, to obtain independent machine cells and balanced machining capacity distribution thoroughout the multi period planning horizon. The total cost function is the sum of total capital investment cost, total idle time cost, total intercellular movement cost and total relocation cost for all the periods in the planning horizon.

Given the notations and the explanations of the options, a mixed integer mathematical programming model for simultaneous part family and machine cell formation under multi period planning horizon, is developed as follows:

Minimize

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\sum_{t=1}^{T}\left[CIM_{k,t}\,Y_{kjt} + IDC_{k,t}\,ID_{kjt}\right] + \sum_{i=1}^{N}\sum_{k=1}^{K_i}\sum_{j=1}^{CN}\sum_{t=1}^{T} AC_{i,t}\,M_{ikjt} +$$

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\sum_{t=1}^{T-1} RLC_{kjt}\left[PC_{kjt} + NC_{kjt}\right]$$

subject to

$$\sum_{j=1}^{CN} Z_{ikjt} = 1 \qquad \forall\, i \in N,\ k \in K_p,\ t \in T \qquad (3.14)$$

$$Z_{ikjt} - Z_{ik-1jt} \le M_{ikjt} \quad \forall\ i\epsilon N\ ,\ k\epsilon K_i\ ,\ k\epsilon LMT_i\ ,\ j\epsilon CN\ ,\ t\epsilon T \qquad (3.15)$$

$$\sum_{i=1}^{N} D_{it}\,t_{ikt}\,Z_{ikjt} + ID_{kjt} = UMC_{kjt}\,Y_{kjt} \quad \forall\ k \in K_i j \in CN, t \in T \qquad (3.16)$$

$$MCL_{jt} \le \sum_{k=1}^{K} Y_{kjt} \le MCU_{jt} \qquad \forall\, j \in CN\ ,\ t \in T \qquad (3.17)$$

$$Y_{kjt-1} - Y_{kjt} = PC_{kjt} - NC_{kjt} \qquad \forall\ k\epsilon K,\ j\epsilon CN,\ t\epsilon T \wedge t<T \qquad (3.18)$$

$$Z_{ikjt},\ M_{ikjt} \ \epsilon\ (0,1) \qquad \forall\ i \in N,\ k \in K_i,\ j \in CN,\ t\epsilon T \qquad (3.19)$$

$$integer\ \ Y_{kjt},\ PC_{kjt},\ NC_{kjt} \qquad \forall\ k \in K,\ j \in CN,\ t \in T \qquad (3.20)$$

The objective of the model is to minimize the various cost functions related with the design of cellular manufacturing systems under multi period planning horizon. There are four cost components considered in the objective function, which are: total capital investment cost, total idle time cost, total intercellular movement cost and total relocation cost for all the periods, in the planning horizon.

The first three components in the total cost function are the modified versions of the ones in Model I, for the multi period planning horizon. They are: total investment cost related with acquiring different types of machines in the system, total idle time cost of the unutilized capacity of each machine type in the system and total intercellular movement cost related with the part movements among the machine cells.

The additional cost component is the relocation cost, which is the result of the changes in the configuration of the machine cells at the end of each period, as a result of the removal of existing machines or installation of new machines in each machine

cell. It assigns a fixed amount of relocation cost, each time a machine is removed from the cell or installed in the cell at the end of each period. Average unit relocation cost for a particular machine type may include installation/removal cost, cost of relocating the oil, water, electrical and material handling systems.

The model developed in this section considers seven different operating constraints related with the design of cellular manufacturing systems under multi period planning horizon.

The first four constraints are the modified versions of the ones in Model I, for the multi period planning horizon. Constraint set (3.14), ensures that each operation of part type i which requires machine type k, is processed in one of the machine cells, in each period. Constraint set (3.15) controls the intercellular movements for each part type among cells, in each period. Constraint set (3.16) ensures that in each machine cell, enough machining capacity is available from each type of machine, to satisfy the production requirements in each period. Constraint set (3.17) brings a lower and an upper limit on the total number of machines in each cell, for each period.

Constraint set (3.18) creates a link among the planning periods, by keeping track of the changes in the number of machine types in each machine cell from period to period. As an analogy, it resembles the inventory balance equations in production and inventory control models. It forces the model to achieve an overall balanced machining capacity distribution, throughout the planning period.

Constraint sets (3.19) and (3.20), ensure the integrality of the model. The model presented above will have (K x CN x T) number of $Y_{k,j,t}$ integer variables, and (K x CN x (T -1)) number of $PC_{k,j,t}$ and $NC_{k,j,t}$ integer variables. For each part type in each period, there will be (CN x $N_{i,t}$) number of $Z_{i,k,j,t}$ binary variables and (CN x (N_i-1)) number of $M_{i,k,j,t}$ binary variables, where K is the number of machine types, CN is the number of machine cells, $N_{i,t}$ is the total number of operations for part type i in period t and T is the total number of periods in the planning horizon.

3.2.4.2. Model IV, Multi Period

Model IV attempts to minimize the overall total cost function for the multi-period planning horizon while simultaneously grouping the machines into machine cells and parts into part families, to obtain independent machine cells and balanced machining capacity distribution throughout the multi period planning horizon. The total cost function is the sum of total capital investment cost, total idle time cost, total intercellular movement cost, total machining cost, total relocation cost and total subcontracting costs, for all the periods in the planning horizon. Given the notations used in the development of the model and the explanation of the possible options, a mixed integer mathematical programming model for simultaneous part family and machine cell, under multi period planning horizon is developed as follows:

Minimize

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\sum_{t=1}^{T}\left[CIM_{k,t}\,Y_{k,j,t}+IDC_{k,t}\,ID_{k,j,t}\right]+\sum_{i=1}^{N}\sum_{k=1}^{K_i}\sum_{j=1}^{CN}\sum_{t=1}^{T}AC_{i,t}\,M_{i,k,j,t}+$$

$$\sum_{k=1}^{K}\sum_{j=1}^{CN}\sum_{t=1}^{T-1}RLC_{k,j,t}\left[PC_{k,j,t}+NC_{k,j,t}\right]+\sum_{i=1}^{N}\sum_{k=1}^{K_{i,t}}\sum_{j=1}^{CN}\sum_{t=1}^{T}D_{i,t}\,t_{i,k,t}\,MC_{k,t}\,Z_{i,k,j,t}$$

$$+\sum_{i=1}^{N}\sum_{t=1}^{T}SC_{i,t}\,D_{i,t}\,S_{i,t}$$

subject to

$$\sum_{j=1}^{CN}Z_{i,k,j,t}\le 1 \qquad \forall\, i\in N,\ k\in K_i,\ t\in T \tag{3.21}$$

$$Z_{i,k,j,t}-Z_{i,k+1,j,t}\le M_{i,k,j,t} \quad \forall\, i\in N,\ k\in K_i,\ k\notin LMT_i,\ j\in CN,\ t\in T \tag{3.22}$$

$$\sum_{i=1}^{N}D_{i,t}\,t_{i,k,t}\,Z_{i,k,j,t}+ID_{k,j,t}=UMC_{k,j,t}\,Y_{k,j,t} \quad \forall\, k\in K_i,\ j\in CN,\ t\in T \tag{3.23}$$

$$MCL_{j,t}\le\sum_{k=1}^{K}Y_{k,j,t}\le MCU_{j,t} \qquad \forall\, j\in CN,\ t\in T \tag{3.24}$$

$$Y_{kj,t+1} - Y_{kj,t} = PC_{kj,t} - NC_{kj,t} \quad \forall\, k \in K,\ j \in CN,\ t \in T \wedge t < T \tag{3.25}$$

$$\frac{1}{N_{i,t}} \left[\sum_{k=1}^{K_i} \sum_{j=1}^{CN} Z_{i,kj,t} \right] + S_{i,t} = 1 \qquad \forall\, i \in N,\ t \in T \tag{3.26}$$

$$Z_{i,kj,t},\ M_{i,kj,t} \in (0,1),\ S_{i,t} \qquad \forall\, i \in N,\ k \in K_i,\ j \in CN,\ t \in T \tag{3.27}$$

$$\textit{integer}\ Y_{kj,t},\ PC_{kj,t},\ NC_{kj,t} \qquad \forall\, k \in K,\ j \in CN,\ t \in T \tag{3.28}$$

The objective of the model is to minimize the various cost functions related with the design of the cellular manufacturing systems under multi period planning horizon. The first four cost components in Model IV are exactly the same as in Model III. The additional fifth and sixth cost components are the total cost of machining the parts in the system and the total cost of subcontracting the parts throughout the multi period planning horizon, respectively.

Model IV operates under eight different operating constraints related with the design of cellular manufacturing systems under multi period planning horizon. Constraint set (3.21) ensures that each operation of the part types may be allocated to the required type of machines in one of the machine cells, in each period. If the part is

not subcontracted, this constraint set prevents the model from allocating the same operations of the part type to different machine cells more than once. Constraint sets (3.22) -(3.25) are exactly the same as in model III, which controls the intercellular movements among cells for each part type in each period, the capacity requirement in each machine cell in each period, the cell size in each period and the link among production planning periods, respectively.

Constraint set (3.26) ensures that each part type is either produced in the system or subcontracted in each period. Constraint sets (3.27) and (3.28) ensure the integrality of the model. The model presented above will have (K x CN x T) number of $Y_{k,j,t}$ integer variables, and (K x CN x (T -1)) number of $PC_{k,j,t}$ and $NC_{k,j,t}$ integer variables. For each part type in each period, there will be (CN x $N_{i,t}$) number of $Z_{i,k,j,t}$ binary variables, (CN x (N_i-1)) number of $M_{i,k,j,t}$ binary variables and one $S_{i,t}$ binary variable, where K is the number of machine types, CN is the number of machine cells, $N_{i,t}$ is the total number of operations for part type i in period t and T is the total number of periods in the planning horizon.

CHAPTER 4

APPLICATION OF GENETIC ALGORITHMS

The mathematical models developed in Chapter 3 work quite efficiently in terms of computational time for small size problems. However, as the problem size becomes larger, the computational time to reach the optimum solution increases exponentially due to the increase in the number of integer variables. Therefore, it is beneficial to develop a heuristic technique to reduce the amount of computational time for large size problems. In this chapter, the application of genetic algorithms as a heuristic technique for obtaining optimal or sub-optimal solutions to the mathematical models developed in Chapter 3 has been investigated. The organization of this chapter is as follows: In section 1, general overview of the commonly used probabilistic search algorithms is given. In section 2, general overview of genetic algorithms is given. In section 3, structure of the genetic operators used in the algorithms and their application by small illustrative example is given. Finally, in section 4 and 5, development and explanation of the genetic algorithms developed for the mathematical models under single and multi period planning horizon is given, respectively.

4.1. Common Probabilistic Search Algorithms

4.1.1. Simulated Annealing

Simulated annealing was first proposed by Kirkpatrick et al. (1983), based on physical annealing, as an effective technique to solve general optimization problems. Physical annealing is a process in which a solid is heated until all particles randomly arrange themselves in the liquid state, followed by a slow cooling process. At each temperature, T, the solid is allowed to reach thermal equilibrium, where energy levels follow Boltzmann distribution (Metropolis et al., 1953). As temperature decreases, the probability tends to concentrate on low energy level states. Careful cooling must be performed, so that temperature is not lowered before thermal equilibrium is reached (Rodrigues and Anjo, 1993).

In simulated annealing, the configuration space of the optimization problem is explored by a controlled hill climbing search in which the control parameter, T, plays the role of the temperature in the physical system (Lam and Delosme, 1988)

The algorithms based on simulated annealing are similar to the local search algorithms. Local search algorithms, in each iteration, search for the best solution by moving from one solution to another within their neighbourhood. The difference between the local search algorithms and simulated annealing is their search mechanism.

In local search algorithms, only the movements that will improve the objective function are allowed, whereas in simulated annealing algorithms, movements that will result in worse objective function values are allowed with a certain probability, in order to avoid to get trapped in a local optimum (Oliveria and Ferreira, 1993). The probability depends on how worse the solution is and on the value of a parameter, the system temperature, based on the previously described analogy.

4.1.2. Tabu Search

Tabu search is introduced by Glover (1989, 1990) as a method of overcoming local optimality. The underlying idea is to forbid some of the search directions to avoid cycling between the same solutions. The algorithm searches the solution space by using a chain of moves from solution to solution, where a move is defined as a transition from a solution to one of its neighbours like in simulated annealing. In each iteration, the tabu search algorithm seeks the "best available move" from the current solution. A best available move is a move which is not included on the tabu list. The tabu list contains the reverse of the last l moves performed. By use of the tabu list, the algorithm avoids returning to the solutions previously visited and increases the chance of leaving the local optimum (Paulli, 1993).

4.2. General Overview

Genetic Algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures -coding of the optimization problem by using a special alphabet- with a structured, yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every iteration, called a *generation*, a new set of string structures is created by using bits and pieces of the old, by using specific "genetic operators" such as reproduction, crossover and mutation. The string structures with poor fitness values (an evaluation of the objective function of the optimization problem), are discarded from the population at the end of each generation, which implies that fitness values of the string structures remaining in the population will either improve or remain the same from generation to generation. The GAs efficiently exploit historical information to speculate on new search points with expected improved performance.

In the literature, three main types of search methods are discussed, which are: Calculus based, Enumerative and Random. Calculus based methods are divided into two parts: indirect and direct. Indirect search methods seek local optima by solving the nonlinear set of equations, whereas the direct search methods seek local optima by hopping on the search function and moving in a direction related to the local gradient.

The major drawback of this method, is being local in shape. i.e. the optima they seek are the best in a neighbourhood of the current point. Starting the search in the neighbourhood of the lower peak will cause the method to miss the higher peak in another neighbourhood.

The main idea of enumerative schemes is to look at objective function values at every point in the search space, one at a time. Although this method is simple and seems attractive, it becomes inefficient for many practical problems where the search space is too large to search one at a time.

The random search algorithms such as, random walk, search the search space and save the best solution. Random search methods, in the long run, can be expected to do no better than enumerative methods in terms of efficiency. GAs behave much more subtly than random search with preservation of the best. Random search ignores the information about the environment accumulated throughout the adaption period, where as GAs use the accumulating information to prune the search space and generate plausible solutions (Austih, 1990).

The difference of GAs from normal optimization and search procedures can be summarized as follows (Goldberg, 1987):

i) GAs work with a coding of the parameter set, not the parameters themselves.

ii) GAs search from a population of points, not a single point.

iii) GAs use fitness (objective function) information, not derivatives or other

auxiliary knowledge.

iv) GAs use probabilistic transition rules, not deterministic rules.

4.3. Structure of Genetic Operators

Genetic algorithms require the natural parameter set of the optimization problem to be coded as a finite length string structure over some finite alphabet. At each iteration, two string structures are chosen from the population at random, and several genetic operators are applied hoping to obtain new string structures with improved fitness values. There are usually three main genetic operators that are being used, which are:

i) Reproduction

ii) Crossover

iii) Mutation

i) Reproduction: It is a process in which string structures for the next generation are chosen from the members of the population at current generation by a probabilistic selection process, which ensures that the expected number of times a string structure is chosen, is approximately proportional to that string structure's fitness value (performance) relative to the rest of the population. Population members with better fitness values thus have a higher probability of contributing one or more offspring to

the next generation. This operator can be seen as an artificial version of natural selection, a Darwinian survival of the fittest among the string structures.

ii) **Crossover:** Under crossover two string structures, randomly selected from the population, exchange portions of their internal representation. Crossover is implemented by choosing a random position in string structure, called the crossover point, and exchanging the segments to the right of this point with another string structure similarly partitioned. Crossover provides new points for further testing within the hyperplanes already in the population.

Figure 4.1 illustrates the application of crossover, where the optimization problem has been coded as a finite length string structure by using a special alphabet. Each position in the string, called *gen*, represents the value of a decision variable for the optimization problem under consideration and each string structure represents a solution, not necessarily optimal, to the optimization problem.

iii) **Mutation:** Under mutation, new string structures are created by occasionally modifying the values of one or more gens in the string structure. Mutation ensures that the probability of searching any region in the problem search space is never zero and works as an insurance policy against the loss of important genetic material information.

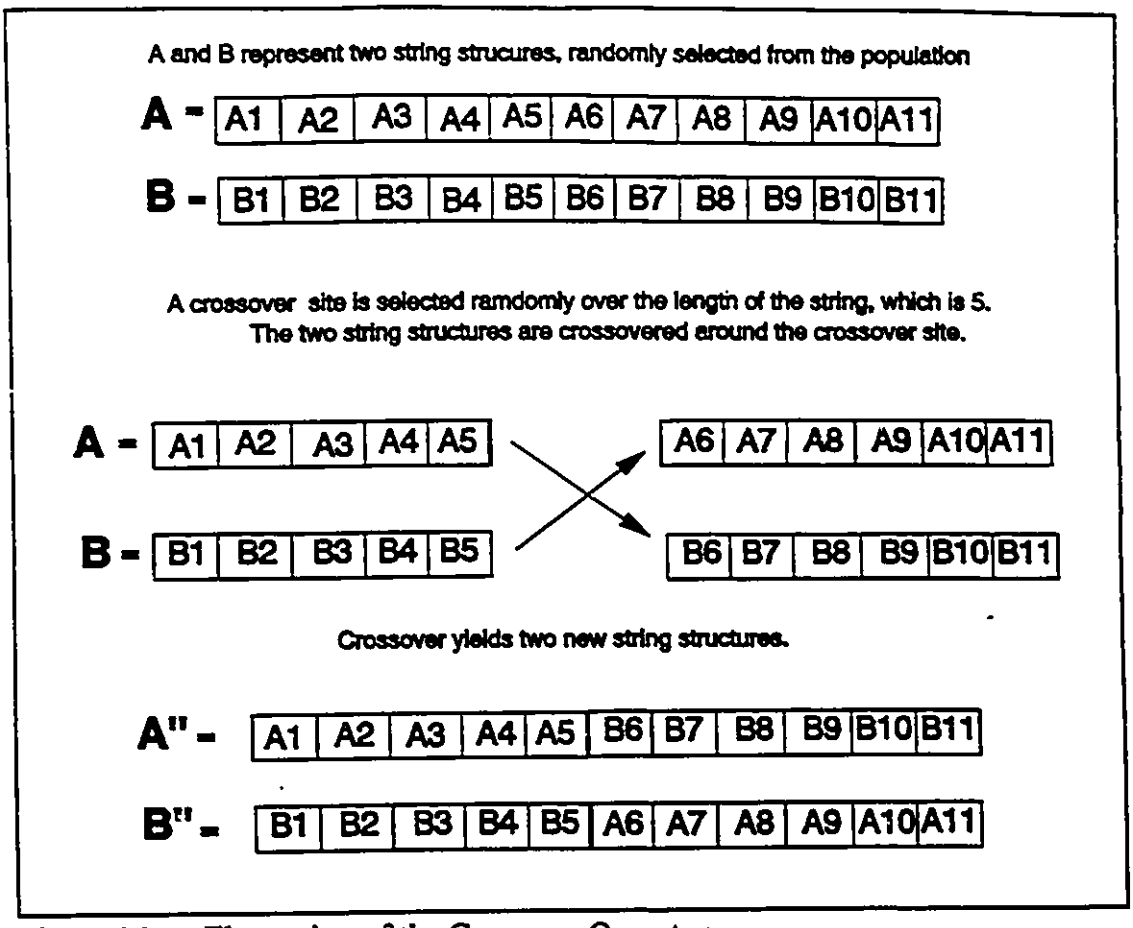

Figure 4.1. Illustration of the Crossover Operator

4.4. Genetic Algorithm Development Under Single Period

In this section, genetic algorithms will be applied to the mathematical models developed under single period planning horizon, in order to find an optimal or sub optimal solution for the mathematical models. The algorithms will simultaneously develop the part families and the machine cells. All the assumptions, objective function

cost components and operating constraints stated for mathematical Model I and Model II, are valid for the genetic algorithm applications. Section 4.3.1 explains the development of GENMOD I, (Genetic Algorithm for Model I) and section 4.3.2. explains the development of GENMOD II, (Genetic Algorithm for Model II). The computer code for the algorithms is in Appendix I.

4.4.1. GENMOD I

In mathematical Model I, the most important decision variable is the binary integer variable $Z_{i,k,j}$ which identifies in which machine cell j, the operation of part type i which requires machine type k will be processed. According to the values of $Z_{i,k,j}$, the machine cells and the part families are formed.

As mentioned before, genetic algorithms code the decision variables of the optimization problem, as a finite length string by using a special alphabet. GENMODI codes the optimization problem in Model I, as a string structure by using binary alphabet where each gen in the string structure corresponds to a particular $Z_{i,k,j}$ variable. Simultaneously, according to the values of $Z_{i,k,j}$ the part families and machine cells are formed. Each string has a length, defined as maxoperation, equal to the multiplication of total number of operations required by all part types by the number of machine cells(CN) specified by the decision maker. Each string gives the information about the

assignment of the operations. For each operation of a particular part type, the number of gens in a string is equal to the number of machine cells (CN). At a given time, only one of these gens will take the value of 1, which identifies that the operation is assigned to that machine cell, and others will take the value of 0, since each operation of a particular part type can be assigned only to one of the machine cells, as specified in the constraint set (3.1). The illustration of the string structure design used in GENMODI is given in figure 4.2.

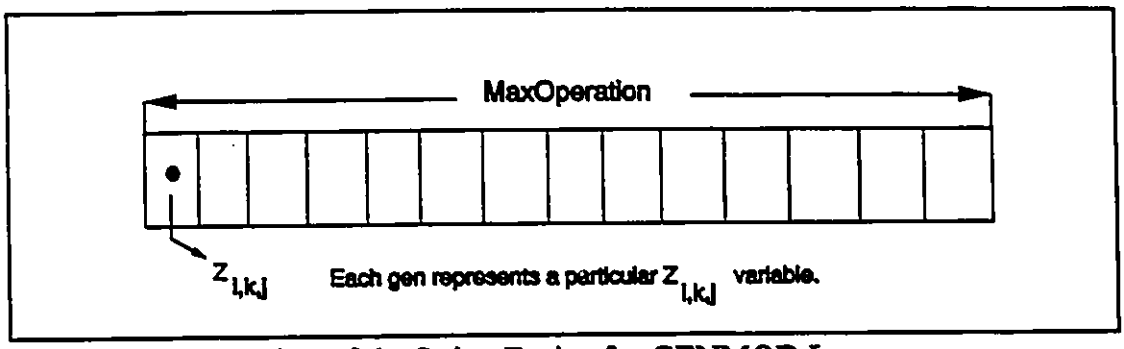

Figure 4.2. Illustration of the String Design for GENMOD I.

GENMOD I defines the population as an array of records. Each individual record, which contains a possible solution to the optimization problem, contains four attributes, which are:

i) String structure (chromosome), which is the coding of the optimization problem as a string structure.

ii) Fitness value (fitness), which is the resulting objective function value calculated by using the cost components of the model under consideration.

iii) Boolean variable (chance), which specifies if the record is eligible to give offspring to next generation. The record is considered as eligible, if its fitness value is less than or equal to the average fitness value of the population.

iv) Real variable (pselect), which is the record's probability of giving offspring to the next generation, i.e. the probability of being selected to the next generation.

The algorithm operates by calling several procedures, which can be summarized as follows:

i) Procedure INIT DATA: Creates the initial records for the first population. For each record, the procedure forms the string structures by randomly assigning operations of each part type to the machine cells and calculating their fitness values.

ii) Procedure CALCULATE STAT: Calculates each individual record's probability of giving offspring to the next generation, which is the attribute *pselect*.

iii) Procedure REPRODUCE: Creates a new population by selecting among the eligible records of the old population and copying them to the new population.

iv) Procedure ARRAY SORT: Sorts the records in the population by using the "Bubble Sort" technique, so that the records having better fitness values (i.e. lower objective function values) will be ranked at higher levels. It also discards the two records with the worst fitness values (i.e. high cost) from the population.

v) Procedure CROSSOVER: Creates two new records, by randomly selecting two records from the current population and mating their string structures.

vi) Procedure FEASIBILITY: Assures that in a string structure for a particular record, each operation is assigned to only one of the machine cells. The procedure removes the duplicated operations, if the same operation is assigned to more than one machine cell.

vii) Procedure PARAMETER CALCULATION: For a given particular record in the population, it calculates the values of the variables Y_{kj} and ID_{kj}.

viii) Procedure MACHINE CHECK: Checks, for a particular record, if the total number of machines in each cell is within cell size limits. The procedure changes the boolean variable (ok) to false, if cell size for at least one of the machine cells is out of the cell size limits.

ix) Procedure RUN LINDO: Calculates the objective function of the record by using the objective function cost components of the model under consideration.

x) Procedure FLIP: Returns the value of 1, if a biased coin results in head with a given probability of head; returns the value of 0, otherwise.

xi) Procedure MUTATION: Alters the value in particular gen with a probability of *pmutation.*

xii) Procedure REPORT: Prepares a report which gives the details about the records in the final population and stores the results in an output file.

The flow chart of GENMOD I is given in figure 4.3. The main steps of the algorithm and detailed explanation of important procedures are as follows:

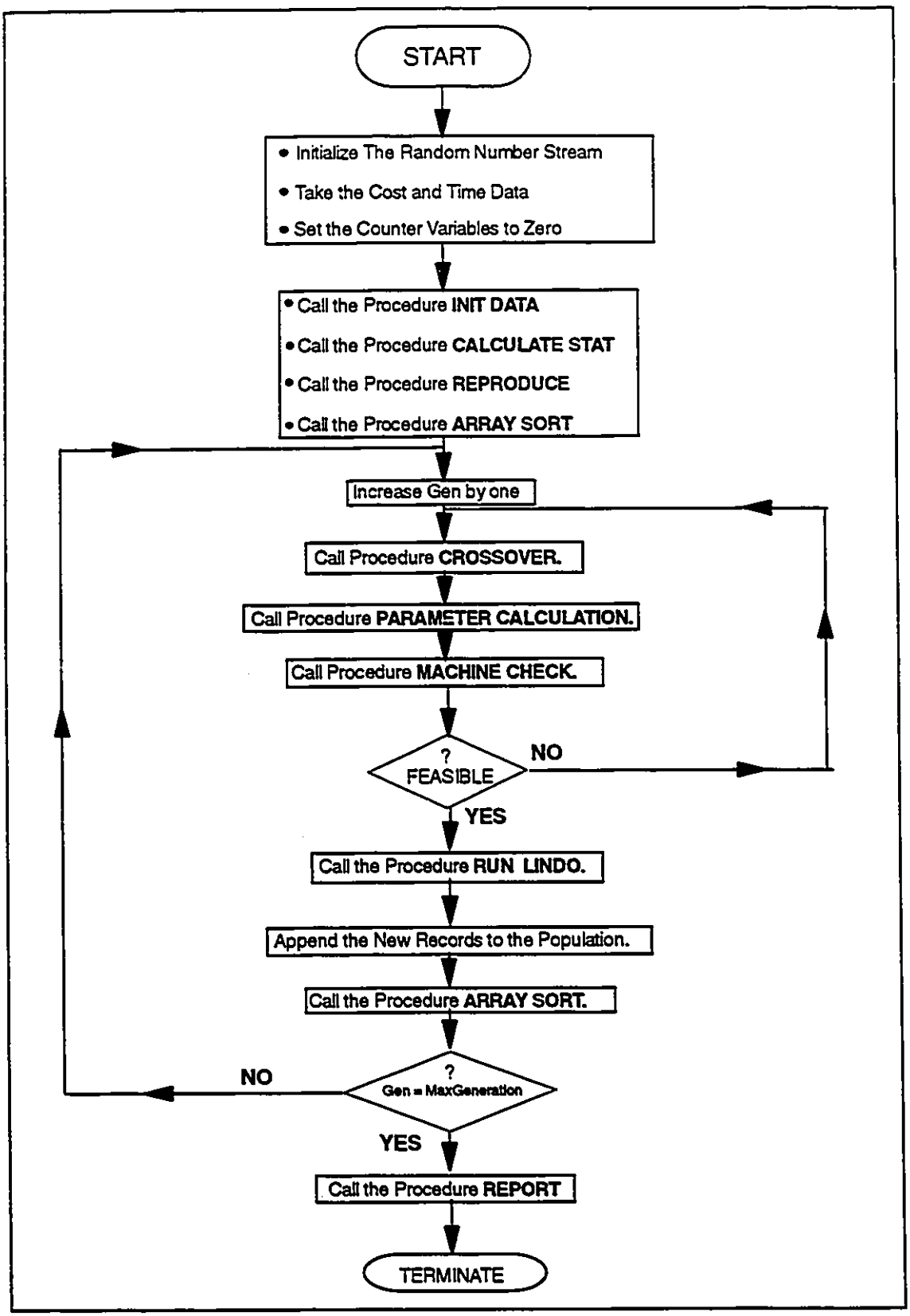

Figure 4.3. Flow Chart of GENMOD I

MAIN ALGORITHM (GENMOD I)

1. Use the system time, as the seed, to initialize the random number stream.
2. Take the time requirement of the parts and the cost data from the input files.
3. Initialize all the counter variables defined below to zero.

   Gen : Counter for the number of generation completed.

   NMutation : Counter for the number of mutations occurred.

   NCrossOver : Counter for the number of crossovers occurred.
4. Call the Procedure INIT DATA, to create string structures for the first population.
5. Call the Procedure CALCULATE STAT, to calculate each individual record's probability of giving offspring to the next generation.
6. Call the Procedure REPRODUCE, to create the new population by selecting among the eligible records in the old population.
7. Call the Procedure ARRAY SORT, to sort the records in the population.
8. Increment Gen by 1.
9. Call the Procedure CROSSOVER, to create two records.
10. Call the Procedure PARAMETER CALCULATION, to calculate the values of the variables $Y_{k,j}$ and $ID_{k,j}$ of the two new records.
11. Call the Procedure MACHINE CHECK, to check if the cell size of the machine cells of the new two records, is within limits.

12. If within limits go to step 13, otherwise discard the two new records and go to step 9.

13. Call the Procedure RUN LINDO, to calculate the fitness values of these two records and append them to the current population.

14. Call the Procedure ARRAY SORT, to update the ranking of the records in the population.

15. If Gen = MaxGeneration then go to step 16, otherwise go to step 8.

16. Call the Procedure REPORT.

17. Terminate.

Procedure INIT DATA

1. Pick the first record in the population.

2. Initialize all the gens in the string structure to zero.

3. For each operation of the part type, beginning from the first part type until the last part type.

    3.1. Randomly select a machine cell (between 1..CN).

    3.2. For each machine cell (from j=1 to CN)

        3.2.1. Calculate the operation location on the string structure.

        3.2.2. If j=selected machine cell then

            Assign the value of 1 to the gen at that location.

Else, assign the value of 0 to the gen at that location.

4. Call the procedure PARAMETER CALCULATION, to calculate the values of Y_{kj} and ID_{kj} for the record.

5. Call the procedure MACHINE CHECK, to check if the cell size for the machine cells of this record is within limits.

6. If within limits then go to step 7, otherwise go to step 2.

7. Call the procedure RUN LINDO, to calculate the fitness value of this record.

8. Are there any records left in the population that has not been initialized? If yes, go to step 9, otherwise go to step 10.

9. Pick the next record in the population and go to step 2.

10. Terminate.

Procedure CALCULATE STAT

1. Initialize the total fitness value, SumFitness, to zero.

2. Calculate the SumFitness by adding up the fitness values of each record in the population.

3. Calculate the average fitness value, Avg, by dividing SumFitness by the size of the population, Pop.

4. For each record in the population.

   4.1. If fitness <= Avg, then Chance = True,

Else, Chance = False.

5. Calculate another total fitness value, SumFitness1, of the records with True Chance attribute.

6. Calculate another average fitness value, Avg1, by dividing SumFitness1 by the number of records with True Chance attribute.

7. Calculate the probability of giving offspring to the next generation, pselect, for each record with a True Chance attribute by using the following formula:

$$Pselect = \frac{2 \times Avg1 - Fitness}{SumFitness1}$$

8. For each record with False Chance attribute, assign the value of 0 to pselect.

9. Terminate.

Procedure REPRODUCE

1. Pick the first record in the population.

2. If the record has a True Chance attribute then

   2.1. Call the procedure FLIP with the probability *pselect*.

   2.2. If FLIP = 1 then, copy the record to the new population.

3. If the new population is full go to step 5, otherwise go to step 4.

4. If the last record in the population has been tried go to step 1, otherwise pick the next record and go to step 2.

5. Terminate.

Procedure CROSSOVER

1. Randomly select two records (Mate 1 and Mate2) from the population for mating.
2. Call the procedure FLIP with the probability of crossover (pcrossover).
3. If FLIP = 1 then, (i.e. crossover will be done)
   - 3.1. Increment counter for crossover, Ncrossover, by one.
   - 3.2. Select a crossover point, crosssite, randomly over the length of the string (1..MaxOperation).
4. If FLIP = 0 then, (i.e. no crossover) assign crosssite to maxoperation.
5. For each gen in the string structure up to crosssite
   - 5.1. Mutate the value of the gen in Mate 1 and assign it to the gen in the new string called Mate 1".
   - 5.2. Mutate the value of the gen in Mate 2 and assign it to the gen in the new string called Mate 2".
6. For each gen in the string structure from crossite to maxoperation
   - 6.1. Mutate the value of the gen in Mate 1 and assign it to the gen in Mate 2".
   - 6.2. Mutate the value of the gen in Mate 2 and assign it to the gen in Mate 1".
7. Call the Procedure FEASIBILITY, to assure that each operation is assigned to

only one machine cell in string structures Mate 1" and Mate 2".

8. Terminate.

Discarding the two worst records from the population, after sorting the population members, at the end of each generation, assures that the average population fitness value will improve or remain the same, from generation to generation.

4.4.2. GENMOD II

In mathematical model II, two important decision variables effect the final solution, which are: $Z_{i,k,j}$ which identifies in which machine cell j, the operation of part type i which requires machine type k will be processed and S_i which identifies if part type i will be subcontracted. According to the values of $Z_{i,k,j}$ and S_i, the machine cells and part families are formed.

GENMOD II codes the optimization problem in Model II, as a string structure by using binary alphabet with a length, defined as MaxOp, equal to the summation of MaxOperation and number of parts in the system (N). Each gen in the string, in the region from 1 to MaxOperation corresponds to a particular $Z_{i,k,j}$ variable and each gen in the string in the region from MaxOperation+1 to MaxOp corresponds to a particular S_i variable. Each string gives the information about the operation assignments and

subcontracting decisions. Simultaneously, according to the values of $Z_{i,k,j}$ and S_i the part families and machine cells are formed. For each operation of a particular part type, the number of gens in a string is equal to the number of machine cells (CN). At a given time, if the part type is not subcontracted, only one of these gens will take the value of 1, which identifies that the operation is assigned to that machine cell, and others will take the value of zero, since each operation of a particular part type can be assigned only to one of the machine cells, as discussed in constraint set 3.7. If the part is subcontracted then all the gens related with the operations of that part will take value of zero, where as, the gen related with the subcontracting decision of that part will take the value of 1, as specified in the constraint set (3.10). The illustration of the string structure design used in GENMOD II is given in Figure 4.4.

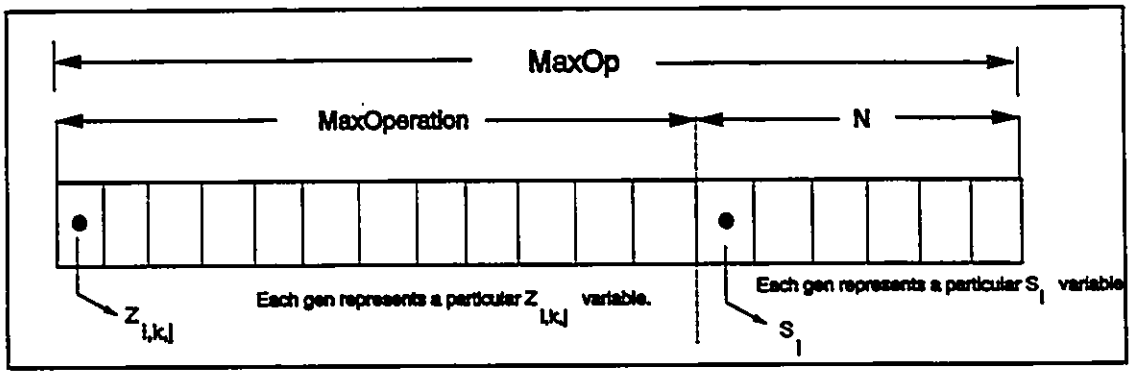

Figure 4.4. Illustration of the String Design for GENMOD II.

GENMOD II, uses the same population structure developed in GENMOD I. The algorithm, in addition to the ones developed in GENMOD I, operates by calling

two new procedures and the modified version of the procedures CROSSOVER and FEASIBILITY. The two new procedures developed for GENMOD II and modified versions of CROSSOVER and FEASIBILITY, can be summarized as follows:

i) Procedure FIND THE BEST: For a given record, the procedure returns the variable *best* which identifies the part type to be subcontracted that will result in maximum savings without violating the machine cell size limits. If subcontracting a particular part type is not feasible, the procedure returns the variable *best* with a value of N+1.

ii) Procedure SUBCONTRACT THE PART: For a given record, the procedure makes the necessary modifications on the string structure for the part type that will be subcontracted (best), such that all the gens related with the operations of that part will take the value of zero and the gen related with the subcontracting decision of that part will take the value of 1.

iii) Procedure CROSSOVER: Creates two new records, by randomly selecting two records from the population and mating only the regions of the string structures related with the assignment of the operations of the parts in the system. The procedure does not modify the regions related with the subcontracting decisions and copy them directly to the new strings(i.e. no information exchange between two strings at this region).

iv) Procedure FEASIBILITY: Assures that in a string structure, if the part type is not subcontracted, each operation of the part type is assigned to only one of the machine

cells. If the part is subcontracted, it assures that none of the operations are assigned to the machine cells in the system.

The first part of the algorithm works exactly the same as in GENMOD I, after the parts are subcontracted the algorithm works for additional number of generations. The algorithm balances the distribution of the capacity by running for additional number of generations after subcontracting, since the distribution of the machining capacity might change after subcontracting. The flow chart of GENMOD II is given in figure 4.5. The main steps of the algorithm and detailed explanation of the procedures FIND THE BEST and modified CROSSOVER, are as follows.

MAIN ALGORITHM (GENMOD II)

1. Use the system time as the seed to initialize the random number stream.
2. Take the time requirement of the parts and the cost data from the input files.
3. Initialize all the counter variables defined below to zero.

   Gen : Counter for the number of generation completed.

   Nmutation : Counter for the number of mutations occurred.

   Ncrossover : Counter for the number of crossovers occurred.
4. Call the Procedure INIT DATA, to create string structures for the first population.
5. Call the Procedure CALCULATE STAT, to calculate each individual record's

probability of giving offspring to the next generation.

6. Call the Procedure REPRODUCE, to create the new population by selecting among the eligible records in the old population.

7. Call the Procedure ARRAY SORT, to sort the records in the population.

8. Increment Gen by 1.

9. Call the Procedure CROSSOVER, to create two new records.

10. Call the Procedure PARAMETER CALCULATION, to calculate the values of the variables Y_{kj} and ID_{kj} of the two new records.

11. Call the Procedure MACHINE CHECK, to check if the cell size of the machine cells of two new records, is within limits.

12. If within limits go to step 13, otherwise discard the two new records and go to step 9.

13. Call the Procedure RUN LINDO, to calculate the fitness values of these two records and append them to the current population.

14. Call the Procedure ARRAY SORT, to update the ranking of the records in the population.

15. If Gen = FinalGeneration then go to step 21, otherwise go to step 16.

16. If Gen = MaxGeneration then go to step 17, otherwise go to step 8.

17. Pick the first record in the population.

18. Repeat .... Until Best = N+1

18.1. Best = N+1.

18.2. Call the Procedure FIND THE BEST.

18.3. If Best<>N+1 then

18.3.1. Call the Procedure SUBCONTRACT THE PART, to make the necessary modifications on the string structure.

18.3.2. Call the Procedure PARAMATER CALCULATION, to calculate the values of $Y_{k,j}$ and $ID_{k,j}$, after subcontracting.

18.3.3. Call the Procedure RUN LINDO.

19. If the last record in the population has been tried go to step 20, otherwise pick the next record and go to step 18.

20. Call the Procedure ARRAY SORT and go to step 8.

21. Call the Procedure REPORT.

22. Terminate.

Procedure FIND THE BEST

1. Assign BestFitness = Current Record's Fitness Value.

2. Pick the first part type in the system.

3. If the part type has been subcontracted already, go to step 10, otherwise go to step 4.

4. Assign the record under consideration to a temporary record and do all the

necessary calculations on that record.

5. Subcontract the part type and assign the value of 0 to all the gens related with the part type's operations and assign the value 1 to the gen related with the subcontracting decision of the part.

6. Call the Procedure PARAMETER CALCULATION, to calculate the new values of the variables $Y_{k,j}$ and $ID_{k,j}$.

7. Call the Procedure MACHINE CHECK, to check if the machine cells in the new string structure are within cell size limits.

8. If within limits, call the Procedure RUN LINDO, to calculate the fitness value of the new string structure, otherwise go to step 10.

9. If the resulting fitness value < BestFitness then assign

   BestFitness = fitness value

   Best = Part type

10. If there exists part types which have not been considered, select the next one and go to step 3, otherwise go to step 11.

11. Terminate.

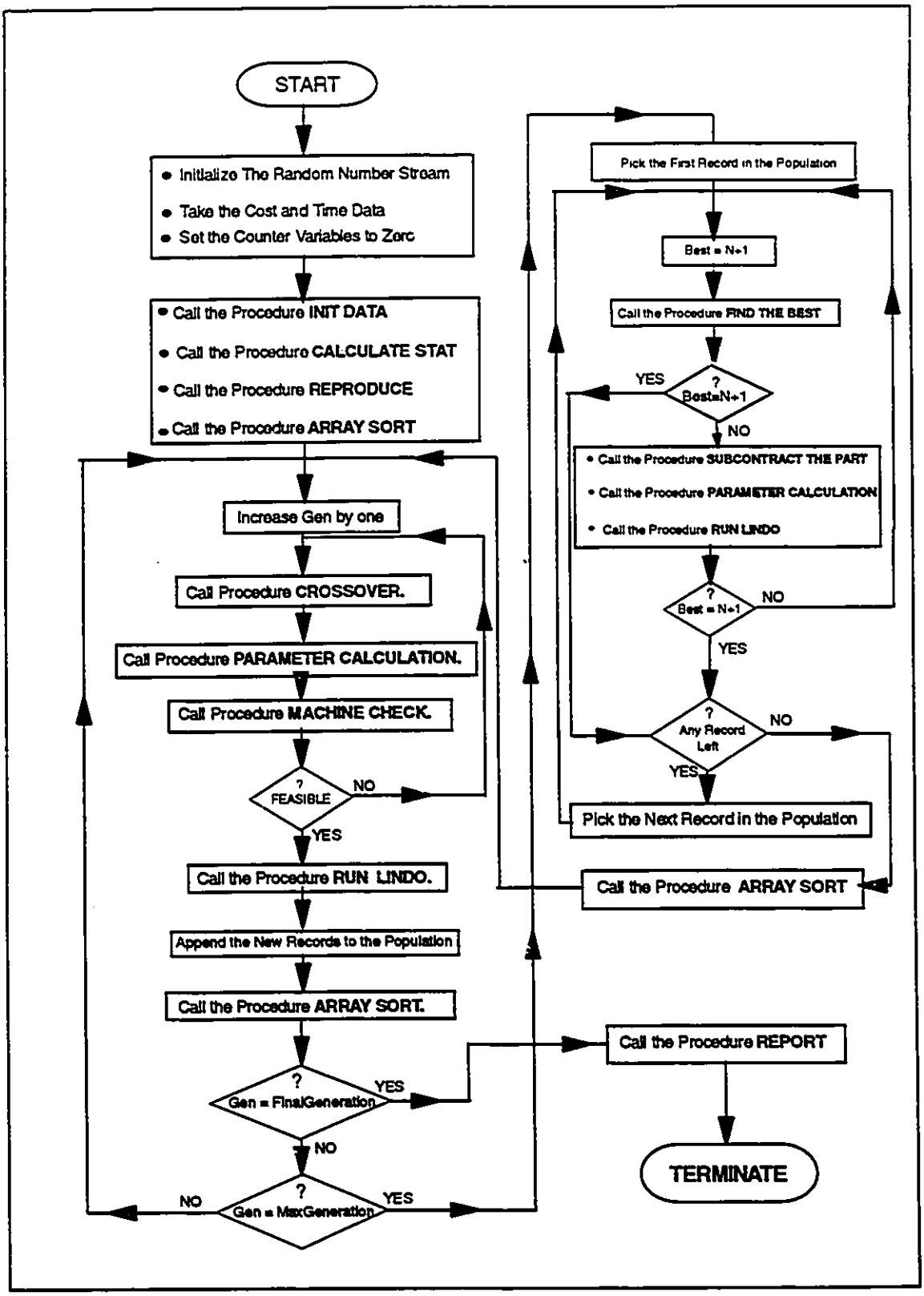

Figure 4.5. Flow Charf of GENMOD II

Procedure CROSSOVER *(modified)*

1. Randomly select two records (Mate 1 and Mate2) from the population for mating.
2. Call the procedure FLIP with the probability of crossover (pcrossover).
3. If FLIP = 1 then, (i.e. crossover will be done)
   - 3.1. Increment counter for crossover, Ncrossover, by one.
   - 3.2. Select a crossover point, crosssite, randomly over the length of the string (1..MaxOperation).
4. If FLIP = 0 then, (i.e. no crossover) assign crosssite to maxoperation.
5. For each gen in the string structure up to crosssite
   - 5.1. Mutate the value of the gen in Mate 1 and assign it to the gen in the new string called Mate 1".
   - 5.2. Mutate the value of the gen in Mate 2 and assign it to the gen in the new string called Mate 2".
6. For each gen in the string structure from crossite to maxoperation
   - 6.1. Mutate the value of the gen in Mate 1 and assign it to the gen in Mate 2".
   - 6.2. Mutate the value of the gen in Mate 2 and assign it to the gen in Mate 1".
7. For each gen in the string structure from maxoperation+1 to maxop
   - 7.1. Copy the value of the gen in Mate 1 and assign it to the gen in Mate 1" without any mutation.

7.2. Copy the value of the string in Mate 2 and assign it to the gen in Mate 2" without any mutation.

8. Call the Procedure FEASIBILITY, to assure that the string structures in Mate1" and Mate 2" are feasible.

9. Terminate.

4.5. Genetic Algorithm Development Under Multi Period

In this section, genetic algorithms will be applied to the mathematical models developed under multi period planning horizon in order to find an optimal or sub optimal solution for the mathematical models. The algorithms will simultaneously develop the part families and machine cells for each period in the planning horizon. All the assumptions, objective function cost components and operating constraints stated for mathematical model III and model IV, are valid for the genetic algorithm applications. Section 4.4.1. explains the development of GENMOD III (Genetic Algorithm for Model III) and section 4.4.2. explains the development of GENMOD IV (Genetic Algorithm for Model IV). The computer code for the algorithms is given in Appendix I.

4.5.1. GENMOD III

In mathematical model III, the most important decision variable is the binary integer variable $Z_{i,k,j,t}$, which identifies in which machine cell j, the operation of part type i which requires machine type k in period t, will be processed. GENMOD III, codes the optimization problem in model III as a [T by MaxOperation] dimensional string structure by using a binary alphabet, where T is equal to the length of the multi period planning horizon and MaxOperation is equal to the multiplication of the total number of operations in the system by the number of machine cells (CN).

Each [1 by Maxoperation] dimensional string structure, exactly the same structure developed in GENMOD I, gives us the information (values of $Z_{i,k,j,t}$) about in which machine cells the operations of the parts will be processed for that period. In each period, for each operation, in a string structure the number of gens is equal to the number of machine cells. At a given time, only one of these gens will take the value of 1 and others will take the value of 0, as specified in constraint set (3.14). The illustration of the string structure design used in GENMOD III, is given in Figure 4.6.

GENMOD III uses the same population structure design developed in GENMOD I, with four attributes which are: string structure for the coding of the optimization problem, fitness value for the optimization problem, chance and probability of selection. The algorithm operates by calling the procedures defined in

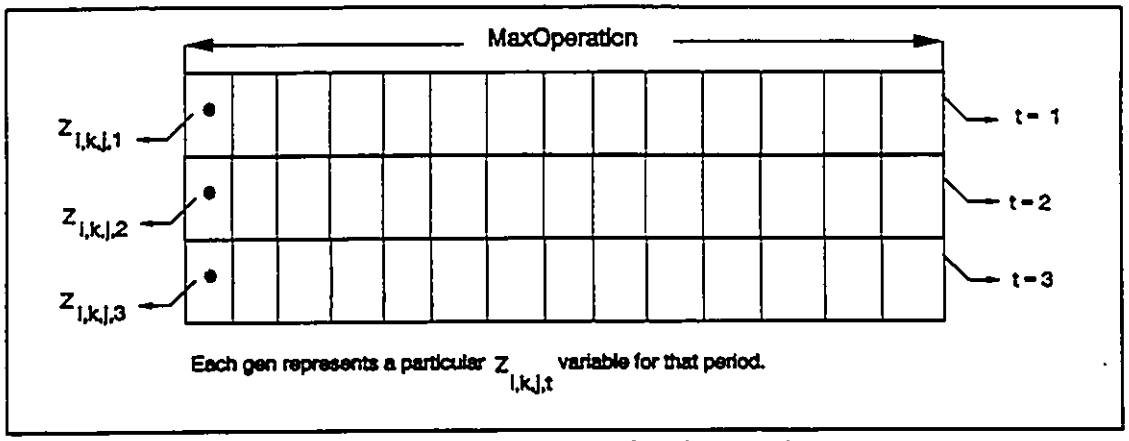

Figure 4.6. Illustration of the String Design for GENMOD III

GENMOD I, where some of the procedures are modified for needs of GENMOD III by adding another variable that will consider the effect of the periods. The summary of the modified procedures are as follows:

i) Procedure INIT DATA: Creates the initial records for the first population. For each record, the procedure forms the string structures by randomly assigning operations of each part type to the machine cells for each period and calculates the fitness values.

ii) Procedure CROSSOVER: Creates two new string structures, by randomly selecting two records from the current population and mating their string structures for a given period.

iii) Procedure FEASIBILITY: Assures that for a given period, in a string structure for a particular record, each operation is assigned to only one of the machine cells. The procedure removes the duplicated operations if the same operation is assigned to more than one machine cell.

iv) Procedure PARAMETER CALCULATION: For a given particular record in the population, it calculates the values of the variables $Y_{k,j,t}$ and $ID_{k,j,t}$ for a given period.

v) Procedure MACHINE CHECK: Checks, for a particular record, if the total number of machines in each cell is within cell size limits for a given period. The procedure changes the boolean variable (ok) to false, if the cell size for at least one of the machine cells is out of the cell size limits.

vi) Procedure RUN LINDO: Calculates the objective function of the record by using the objective function cost components of the model under consideration.

vii) Procedure REPORT: Prepares a report which gives the details about the records in the final population and stores the results in an output file by considering all the periods.

The flow chart of GENMOD III is given in figure 4.7. The main steps of the algorithm are as follows:

MAIN ALGORITHM (GENMOD III)

1. Use the system time as the seed to initialize the random number stream.
2. Take the time requirement of the parts and the cost data from the input files.
3. Initialize all the counter variables defined below to zero.

   Gen : Counter for the number of generation completed.

   Nmutation : Counter for the number of mutations occurred.

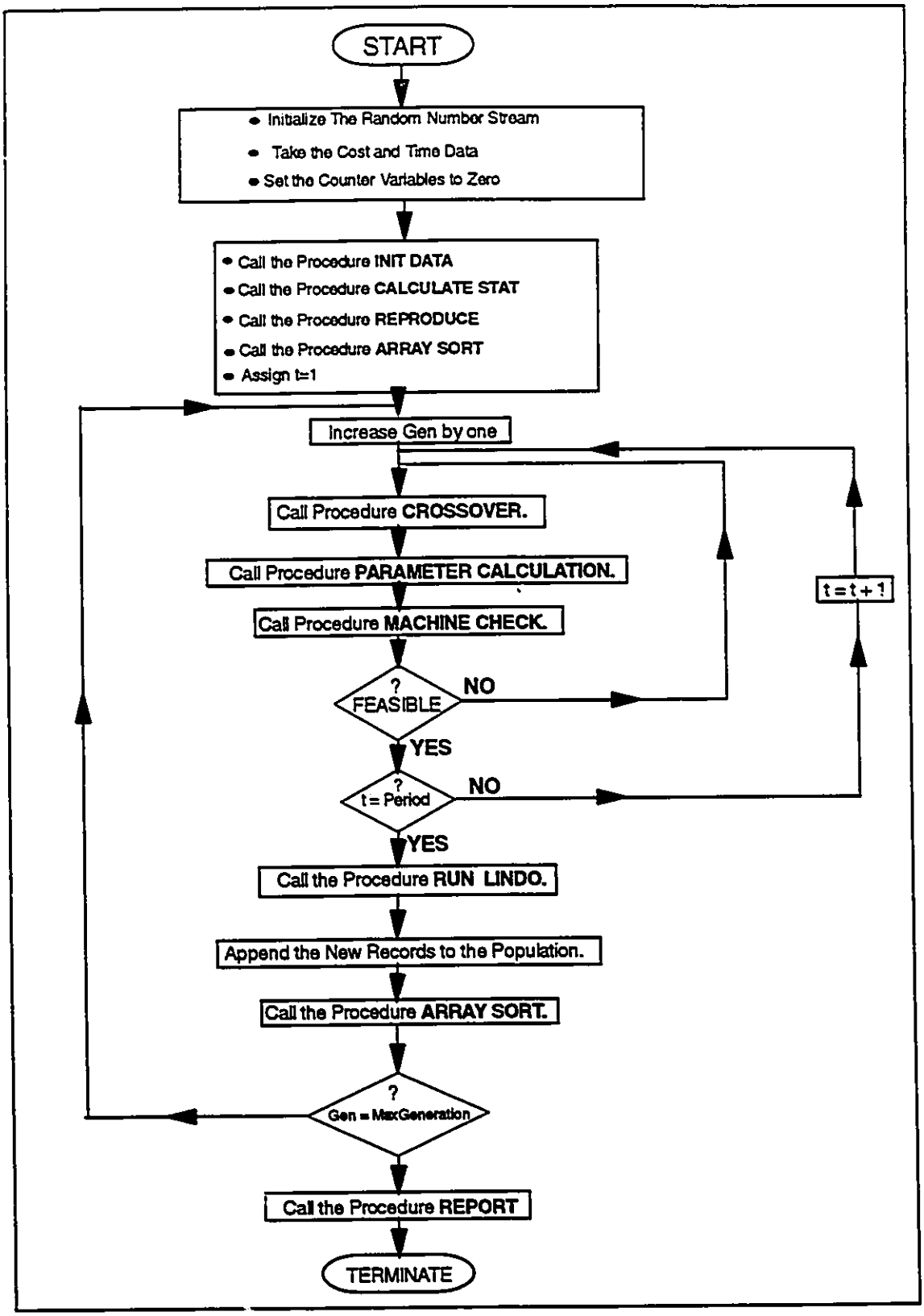

Figure 4.7. Flow Chart of GENMOD III

Ncrossover : Counter for the number of crossovers occurred.

4. Call the Procedure INIT DATA, to create string structures for the first population.

5. Call the Procedure CALCULATE STAT, to calculate each individual record's probability of giving offspring to the next generation.

6. Call the Procedure REPRODUCE, to create the new population by selecting among the eligible records in the old population.

7. Call the Procedure ARRAY SORT, to sort the records in the population.

8. Increment Gen by 1.

9. For each period in the planning horizon (t=1..T)

   9.1. Call the Procedure CROSS OVER, to create two new string structures for the period under consideration.

   9.2. Call the Procedure PARAMETER CALCULATION, to calculate the values of the variables $Y_{k,j,t}$ and $ID_{k,j,t}$ of the two new string structures for the period under consideration.

   9.3. Call the Procedure MACHINE CHECK, to check if the cell size of machine cells of two new strings are within limits.

10. If all the machine cells for all the periods are within limits then go to step 11, otherwise discard the two new records and go to step 9.

11. Call the Procedure RUN LINDO, to calculate the fitness values of the two new

records and append them to the current population.

12. Call the Procedure ARRAY SORT, to update the ranking of the records in the population.

13. If Gen = MaxGeneration then go to step 14, otherwise go to step 8.

14. Call the Procedure REPORT.

15. Terminate.

4.5.2. GENMOD IV

In mathematical model IV, two important decision variables effect the final solution, which are: $Z_{i,k,j,t}$ which identifies in which machine cell j, the operation of part type i which requires machine type k in period t, will be processed and $S_{i,t}$ which identifies if part type i will be subcontracted in period t. According to the values of $Z_{i,k,j,t}$ and $S_{i,t}$, the machine cells and part families are formed. GENMOD IV, codes the optimization problem in model IV as a [T by MaxOp] dimensional string structure by using a binary alphabet, where T is equal to the length of the multi period planning horizon and MaxOp is equal to the summation of MaxOperation and total number of parts in the system (N).

In each [1 by MaxOp] dimensional string structure, the same structure design developed in GENMOD II, the gens in the region from 1 to MaxOperation corresponds

to a particular $Z_{i,k,j,t}$ variable and the gens in the region from MaxOperation+1 to MaxOp corresponds to a particular $S_{i,t}$ variable. Simultaneously, according to the values of $Z_{i,k,j,t}$ and $S_{i,t}$, the part families and the machine cells are formed. In each period, for each operation of a particular part type, the number of gens in a string is equal to the number of machine cells (N). At a given time, if the part type is not subcontracted, only one of these gens will take the value of 1 and the others will take the value of 0, as specified in constraint set (4.21). If the part type is subcontracted for that period, all the gens related with the operations of that part type will take the value of 0, where as, the gen related with the subcontracting decision of that part will take the value of 1, as specified in constraint set (4.26). The illustration of the string design used in GENMOD IV, is given in Figure 4.8.

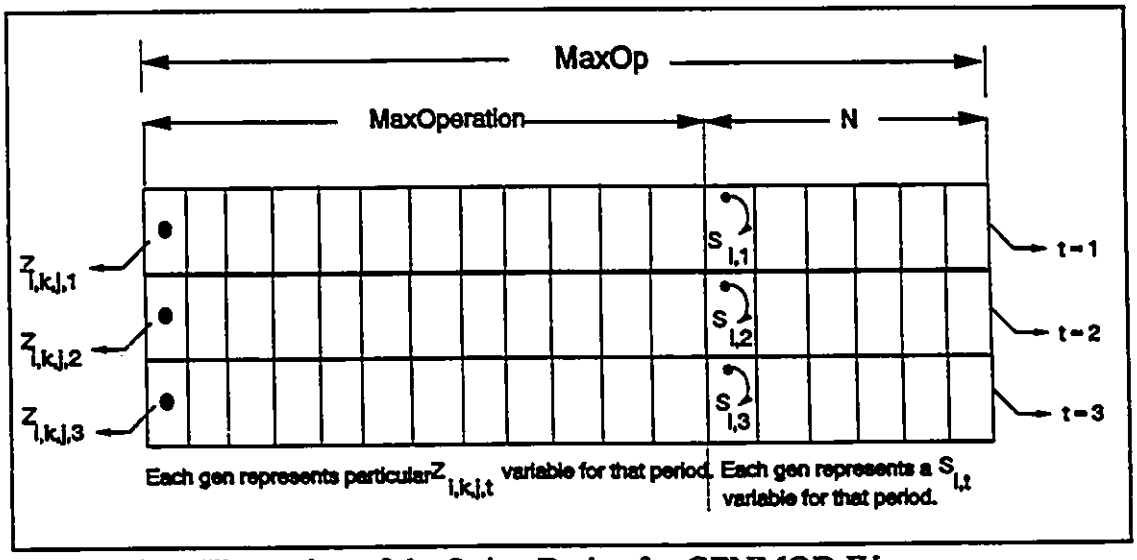

Figure 4.8. Illustration of the String Design for GENMOD IV.

GENMOD IV uses the same population structure design developed in GENMOD I. The algorithm, in addition to the ones used in GENMOD III, operates by calling the modified versions of the procedures FIND THE BEST, SUBCONTRACT THE PART, CROSSOVER and FEASIBILITY. These modified procedures can be summarized as follows:

i) Procedure FIND THE BEST: For a given record, the procedure returns the variable *best* which identifies the part type to be subcontracted that will result in maximum savings without violating the machine cell size limits for a given period. If the subcontracting a particular part type is not feasible in that period, the procedure returns the variable *best* with a value of N+1.

ii) Procedure SUBCONTRACT THE PART: For a given record, the procedure makes the necessary modifications on the string structure for the part type that will be subcontracted (*best*) for the period under consideration, such that all the gens related with the operations of that part will take the value of zero and the gen related with the subcontracting decision of that part will take the value of 1.

iii) Procedure CROSSOVER: Creates two new records, by randomly selecting two records from the population and mating only the regions of the string structures related with the assignment of the operations of the parts in the system for a given period. The procedure does not modify the regions related with the subcontracting decisions and copy them directly to the new strings(i.e. there is no information exchange between two

strings at this region).

iv) Procedure FEASIBILITY: Assures that in a string structure, if the part type is not subcontracted in a given period, each operation of the part type is assigned to only one of the machine cells. If the part is subcontracted in a given period, it assures that none of the operations are assigned to the machine cells in the system.

The flow chart of GENMOD IV is given in Figure 4.9. The main steps of the algorithm are as follows:

MAIN ALGORITHM (GENMOD IV)

1. Use the system time as the seed to initialize the random number stream.
2. Take the time requirement of the parts and the cost data from the input files.
3. Initialize all the counter variables defined below to zero.

   Gen : Counter for the number of generation completed.

   NMutation : Counter for the number of mutations occurred.

   NCrossOver : Counter for the number of crossovers occurred.
4. Call the Procedure INIT DATA, to create string structures for the first population.
5. Call the Procedure CALCULATE STAT, to calculate each individual record's probability of giving offspring to the next generation.
6. Call the Procedure REPRODUCE, to create the new population by selecting

among the eligible records in the old population.

7. Call the Procedure ARRAY SORT, to sort the records in the population.

8. Increment Gen by 1.

9. For each period in the planning horizon (t=1..T)

   9.1. Call the Procedure CROSS OVER, to create two new string structures for the period under consideration.

   9.2. Call the Procedure PARAMETER CALCULATION, to calculate the values of the variables $Y_{k,j,t}$ and $ID_{k,j,t}$ of the two new string structures for the period under consideration.

   9.3. Call the Procedure MACHINE CHECK, to check if the cell size of machine cells of two new strings is within limits.

10. If all the machine cells for all the periods are within limits then go to step 11, otherwise discard the two new records and go to step 9.

11. Call the Procedure RUN LINDO, to calculate the fitness values of the two new records.

12. Call the Procedure ARRAY SORT, to update the ranking of the records in the population.

13. If Gen = FinalGeneration then go to step 19, otherwise go to step 14.

14. If Gen = MaxGeneration then go to step 15, otherwise go to step 8.

15. Pick the first record in the population.

16. For each period in the planning horizon (t=1..T),

Repeat...Until Best = N+1.

16.1. Best = N+1.

16.2. Call the Procedure FIND THE BEST.

16.3. If Best<>N+1 then

16.3.1. Call the Procedure the SUBCONTRACT THE PART, to make the modification on the string structure for the part to be subcontracted.

16.3.2. Call the Procedure PARAMETER CALCULATION, to calculate the new values of $Y_{k,j,t}$ and $ID_{k,j,t}$ after the modification.

16.3.3. Call the Procedure RUN LINDO.

17. If the last record in the population has been tried go to step 18, otherwise pick the next record and go to step 16.

18. Call the Procedure ARRAY SORT and go to step 8.

19. Call the Procedure REPORT.

20. Terminate.

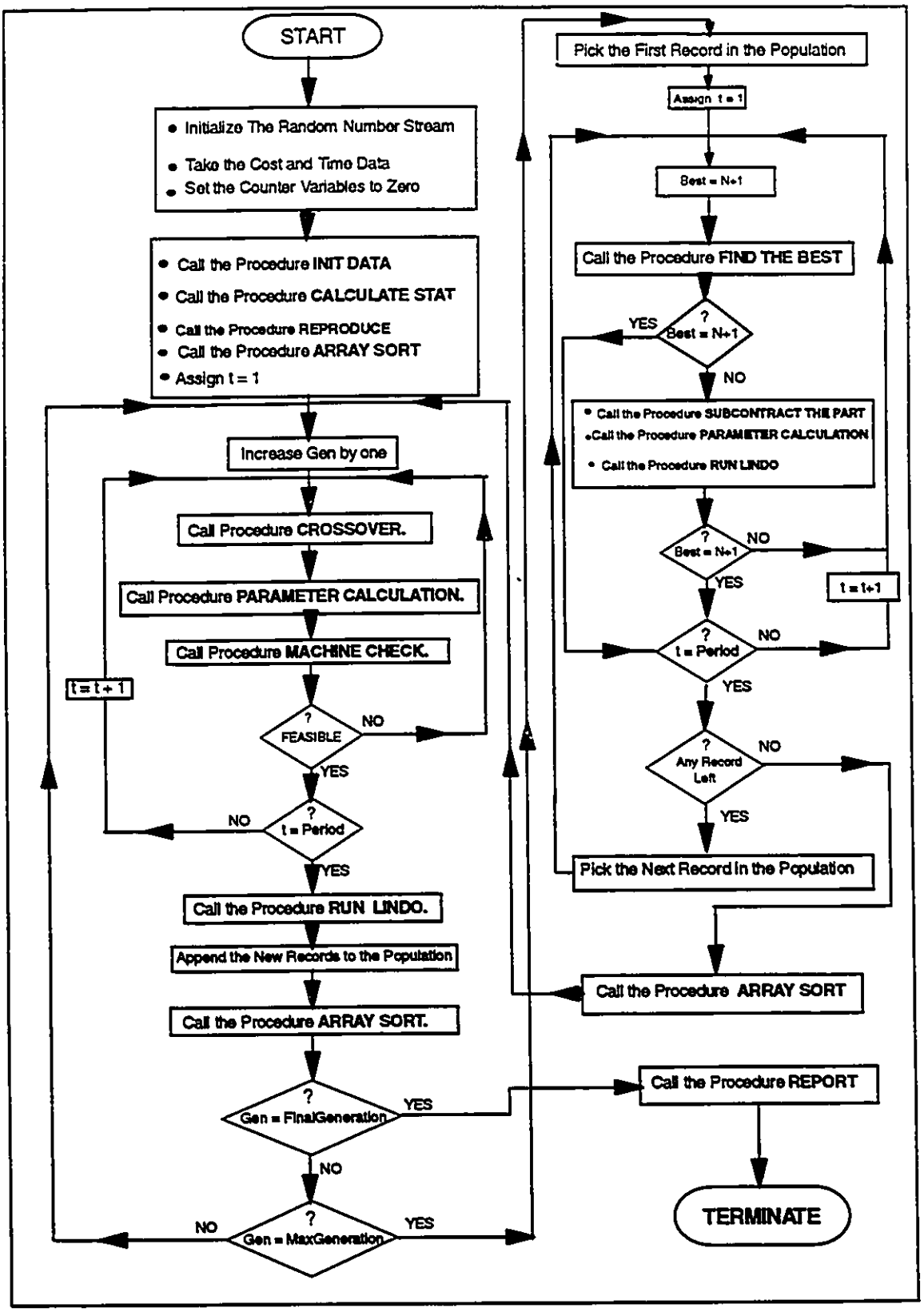

Figure 4.9. Flow Chart of GENMOD IV

CHAPTER 5

NUMERICAL RESULTS

In this chapter the mathematical models developed under single and multi period planning horizons are tested by using several numerical examples of different sizes. A different type of multi period planning strategy is developed and its efficiency in terms of arising cost values and machine cell utilizations is tested with Models III and IV. Furthermore, applicability and efficiency of Genetic Algorithms is tested by using a large size example and the results are compared with the mathematical models. A computer code is developed by using the Turbo Pascal language to create the Lindo input files for testing purposes. The computer code is in Appendix II. The models are run by using hyper Lindo on IBM compatible 486 PC and the results are analyzed.

5.1.1. Model I, Single Period

In this section, Model I is tested by using two examples of different sizes and the results are analyzed. Cost trade-offs among intercellular movement and machine duplication are under consideration. The first example considers 10 different part types and 6 different machine types, and allows two machine cell formation, whereas the second model considers 15 different part types, 10 different types of machines and allows three machine cell formation. For each example, 2000 machine hour capacity is assigned to each of the machines and a cost of $ 2000 is charged for each intercellular movement occurred in the system.

5.1.1. Case I, Two Machine Cell Formation

In this example two machine cell formation is considered, having 6 different type of machines, 10 different parts types under consideration with a minimum annual demand of 12000 units, the model creates the machine cells and allocates the operations of 200000 parts to the machines. If it is economically feasible, the model may assign some of the operations of the parts in other cells and create intercellular movements. Machine requirements and annual demand of the parts for the hypothetical example are given in Table 5.1. In addition, various cost parameters used during the calculations

are given in Table 5.2

Table 5.1. Machine Requirements and Annual Demand of the Parts

Part	Machines						Annual Demand
	1	2	3	4	5	6	
1	2	4	0	3	0	0	12000
2	0	3.2	0	1.8	0	1.7	24000
3	0	0	2.2	0	3.5	0	36000
4	0	2	0	2.3	0	3	18000
5	3	2.6	0	0	0	2.8	20000
6	0	1.9	2.1	0	2.4	0	14000
7	0	0	2.5	0	2.7	0	18000
8	2.6	2.9	0	2.4	0	2.2	16000
9	0	0	2.0	0	2.2	2	20000
10	3	3.2	2.8	2	0	0	20000

(Entries in the table are the unit processing times in minutes.)

Table 5.2. Machining, Capital and Idle Time Costs

Machine Type	Machining Cost ($/hr)	Capital Costs($/year)	Idle Time Costs($/hr)
1	9.0	16,000	4.0
2	11.0	20,000	6.0
3	10.0	18,000	5.0
4	11.0	24,000	6.0
5	7.0	14,000	3.0
6	8.0	17,000	4.0

The results of Model I, for two cell formation, indicate that the first machine cell and the second machine cell contains at least one unit of machine from each machine type. Additionally, the first machine cell contains two units of type 2 machine and the second machine cell contains two units of type 3 and type 5 machines. On the other hand, the first machine cell contains the part family which consists of part types 1, 2, 4, 6, 8 and 9, and the second machine cell contains the part family which consists of part types 3, 4, 5, 7 and 10. Part type 4 is the exceptional part type, its first operation was allocated to machine type 2 in machine cell 1, and the rest of the operations were allocated to machine types 4 and 6 in machine cell 2. The model allocated the first operation of part type 4 on machine type 2 in machine cell 1, instead of duplicating the machine type 4 in machine cell 2, since allocating the operation in machine cell 1 and creating intercellular movements is cheaper than the cost of duplicating the machine type 2 in machine cell 2. The final results of the part families and the machining cells are summarized in Table 5.3. and Table 5.4., respectively. Also various cost results of the final optimal solution are given in Table 5.5.

Table 5.3. Final Part Family Composition

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,2,4,6,8,9	6
Cell 2	3,4,5,7,10	5

Table 5.4. Final Machining Cell Composition

Machining Cell	Machine Type						Total Number of Machines
	1	2	3	4	5	6	
Cell 1	1	2	1	1	1	1	7
Cell 2	1	1	2	1	2	1	8

(Entries in the table represent the number of machines)

Table 5.5. Final Cost Values

Cost Function	Cost Value ($)
Capital Investment Cost	270,000
Idle Time Cost	24,301
Intercellular Movement Cost	2,000
Total Cost	**296,301**

5.1.2. Case II, Three Machine Cell Formation

In this section the model is tested by using a bigger size example. The example used has 15 different part types and 10 different machine types, and considers three machine cell formation. The model groups the machines into three machine cells and allocates the operations of 250000 parts. Machine requirements and annual demand of the parts for the hypothetical example are given in Table 5.6. In addition, various cost parameters used during the calculations are given in Table 5.7.

Table 5.6. Machine Requirements and Annual Demand of the Parts

Part Type	Machine Types										Annual Demand
	1	2	3	4	5	6	7	8	9	10	
1	0	0	1.8	1.9	0	2.2	0	0	0	0	15000
2	0	0	0	0	2.4	0	2.0	0	0	0	15000
3	0	0	0	0	0	0	0	2.3	0	2.0	17000
4	0	0	0	0	1.4	0	2.2	2.0	0	1.6	18000
5	0	3.2	0	3.1	0	0	0	0	2.4	0	16000
6	2.6	2.4	0	0	0	0	0	0	2.0	0	17000
7	0	0	0	3.0	2.0	0	2.0	0	0	0	16000
8	0	0	1.7	1.8	0	2.8	0	0	0	0	16000
9	2.8	1.3	0	0	0.8	0	0	0	0	0	20000
10	0	0	0	0	1.8	0	1.8	0	0	0	16000
11	0	0	0	0	1.9	0	2.2	2.1	0	1.5	16000
12	0	0	0	3.2	0	0	1.7	0	0	0	18000
13	0	0	2.9	0	0	0	0	0	1.8	0	16000
14	0	0	0	0	2.0	0	0	0	0	1.5	17000
15	0	0	0	0	2.1	0	1.6	0	0	0	17000

(Entries in the table are the processing time in minutes.)

Table 5.7. Machining, Capital and Idle Time Costs

Machine Type	Machining Cost($/hr)	Capital Cost($/year)	Idle Time Cost($/hr)
1	9	25000	4
2	12	26000	5
3	10	24000	4
4	9	27000	4
5	8	28000	3
6	11	29000	5
7	7	23000	3
8	9	25000	4
9	8	27000	3
10	10	28000	4

The results indicate that the first machine cell consists of machine types 1,2,3,4,6 and 9, the second machine cell consists of machine types 5,7,8 and 10, and the third machine cell consists of machine types 4,5 and 7. Also the first machine cell consists of part types 1,5,6,8,9 and 13, the second machine cell consists of part types 3,4,10,11, and 14 and finally the third machine cell consists of the part types 2,7,9,12 and 15. From the results it is seen that there exists an exceptional part type, part type 9, in the system. The model allocated the first two operations of part type 9 in machine cell 1 and the last one in machine cell 3 on machine type 5. The model did not duplicate machine type 5 in machine cell 1 since the cost of intercellular movement is less than the machine duplication and excess unutilized capacity cost. The final part family and the machine cell formations are summarized in Table 5.8. and Table 5.9, respectively. Also various cost results of the final optimal solution are given in Table 5.10.

Table 5.8. Final Part Family Composition

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,5,6,8,9,13	6
Cell 2	3,4,10,11,14	5
Cell 3	2,7,9,12,15	5

Table 5.9. Final Machining Cell Composition

Manufacturing Cell	Machine Type										Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	
Cell 1	1	1	1	1	0	1	0	0	1	0	6
Cell 2	0	0	0	0	1	0	1	1	0	1	4
Cell 3	0	0	0	1	1	0	1	0	0	0	3

(Entries in the table represent the number of machines)

Table 5.10. Final Cost Values

Cost Function	Cost Value ($)
Capital Investment Cost	340,000
Idle Time Cost	11,113
Intercellular Movement Cost	2,000
Total Cost	**353,113**

5.2. Model II, Single Period

In this section, Model II is tested by using two examples developed in section 5.1. Cost trade-offs among intercellular movements, machine duplication and subcontracting are under consideration.

5.2.1. Case I, Two Machine Cell Formation

The model is tested by using the same example used to test Model I; 10 different part types and 6 different machine types are under consideration. Machine requirements and annual demand of the part types are the same as given in Table 5.1. The cost parameters used, in addition the ones in Table 5.2, in the development of the model are given in Table 5.11.

Table 5.11. Subcontracting Costs

Part Type	Subcontracting Cost ($/unit)	Part Type	Subcontracting Cost ($/unit)
1	5	6	5.8
2	4.9	7	4.7
3	4.2	8	5.4
4	0.45	9	4.7
5	5	10	5.8

The list of the LINDO input files are in Appendix III. The results indicate that the first machine cell and the second machine cell contains at least one unit of machine from each machine type. Additionally, the first machine cell contains two units of machine type 5 and the second machine cell contains two units of type 2 and type 3 machines. On the other hand, the first machine cell contains the part family which consists of part types 1,3,8 and 9, and the second machine cell contains the part family

which consists of part types 2,5,6,7 and 10. Part type 4 is subcontracted. The part family composition is different than the one obtained under Model I. Since part type 4 is subcontracted, the model reallocates the machines and the operations to achieve a balanced distribution of the workload. The model subcontracts the whole demand of part type 4 based on cost trade-offs since the cost of producing part type 4 in the system, is higher than the cost of subcontracting. The final part family and machine cell compositions are summarized in Table 5.12. and Table 5.13, respectively. Also, various cost values of the final optimal solution are given in Table 5.14.

Table 5.12. Final Part Family Composition

Machining Cell	Part Families	
	Part Type	Total
Cell 1	1,3,8,9	4
Cell 2	2,5,6,7,10	5

Table 5.13. Final Machining Cell Composition

Machining Cell	Machine Type						Total Number of Machines
	1	2	3	4	5	6	
Cell 1	1	1	1	1	2	1	7
Cell 2	1	2	2	1	1	1	8

(Entries in the table represent the number of machines)

Table 5.14. Final Cost Values

Cost Function	Cost Value ($)
Capital Investment Cost	270,000
Idle Time Cost	35,641
Intercellular Movement Cost	0
Machining Cost	208,167
Subcontracting Cost	8000
Total Cost	**521,808**

5.2.2. Case II, Three Machine Cell Formation

The model is tested by using the same example used to test Model I; 15 different part types and 10 different machine types are under consideration. The machine requirements and the annual demand of the parts are the same as given in Table 5.6. The cost parameters used in development of the model, in addition to the ones in Table 5.7 are given in Table 5.15.

Table 5.15. Subcontracting Costs ($/unit)

Part Type	Value	Part Type	Value	Part Type	Value
1	2.2	6	2.4	11	2.1
2	1.8	7	1.9	12	1.7
3	2.4	8	2.3	13	2.0
4	2.7	9	0.45	14	2.2
5	2.5	10	2.1	15	1.6

The results indicate that the first machine cell consists of machine types of 1,2,3,4,6 and 9, the second machine cell consists of machine types of 5,7,8 and 10, and the third machine cell consists of machine types of 4,5 and 7. The same machine cell formation is obtained, as given in Table 5.13. The first machine cell consists of part types 1,5,6,8 and 13, the second machine cell consists of part types 3,4,10,11, and 14, and finally the third machine cell consists of part types 2,7,12 and 15. Part type 9 is subcontracted. The model subcontracts part type 9 based on cost trade-offs, since the cost of producing part type 9 within the system is higher than the cost of subcontracting the part. The final composition of the machine cells remained the same as given in Table 5.13. The final composition of the part families is changed and given in Table 5.16. The final cost values of the optimal solution are given in Table 5.17.

Table 5.16. Final Part Family Composition

Machining Cell	Part Families	
	Part Type	Total
Cell 1	1,5,6,8,13	5
Cell 2	3,4,10,11,14	5
Cell 3	2,7,12,15	4

Table 5.17. Final Cost Values

Cost Function	Cost Value ($)
Capital Investment Cost	340,000
Idle Time Cost	17,808
Intercellular Movement Cost	0
Machining Cost	192,484
Subcontracting Cost	9,000
Total Cost	**559,292**

5.3. Model III, Multi Period

In this section, Model III is tested by using an example that includes 10 different part types, 7 different machine types and the results are analyzed. The objective is to simultaneously develop the part family and machine cell configurations for each period in the planning horizon while attempting to minimize the exceptional parts and obtain a balanced machining capacity distribution throughout the planning horizon. Cost trade-offs among intercellular movement, machine duplication and machine relocation for each period, are under consideration.

The length of the planning period is taken as 3 years. For each period in the planning period, 2 machine cell formation is considered and 2000 machine hour capacity is assigned to each of the machines in the machine cells, $2,000 for each

intercellular movement and $5,000 for each machine relocation (installation or removal) is charged respectively, during the calculations. Machine requirements and annual demand of the parts during the planning horizon are given in Tables 5.18 and 5.19, respectively. The cost parameters used in the example, given in Table 5.20, are assumed to be constant throughout the planning horizon.

Table 5.18. Machine Requirements of the Parts

Part Type	Machines						
	1	2	3	4	5	6	7
1	2.0	0	1.9	0	1.4	0	0
2	0	0	0	1.7	0	2.2	1.9
3	0	0	0	2.1	0	0	2.0
4	0	1.8	0	2.3	0	0	0
5	0	1.5	0	2.2	0	0	0
6	0	2.5	0	0	1.9	0	1.8
7	0	0	0	2.0	0	1.7	0
8	2.1	0	2.4	0	0	0	0
9	1.7	0	0	2.4	0	0	0
10	0	2.7	0	0	1.8	1.5	0

Table 5.19. Part Demand for Each Period

Part Type	Period 1	Period 2	Period 3
1	21,000	22,000	21,000
2	19,000	16,000	13,000
3	20,000	18,000	14,000
4	20,000	21,000	18,000
5	22,000	23,000	19,000
6	23,000	9,000	6,000
7	22,000	19,000	15,000
8	19,000	20,000	21,000
9	18,000	20,000	15,000
10	20,000	9,000	7,000

Table 5.20. Capital, Machining and Idle Time Costs

Machine Type	Capital Cost ($/year)	Machining Cost ($/hr)	Idle Time Cost ($/hr)
1	18,000	9.0	4.0
2	20,000	12.0	6.0
3	22,000	11.0	4.0
4	24,000	12.0	3.0
5	23,000	11.0	4.0
6	19,000	10.0	3.0
7	20,000	9.0	5.0

The resulting part family and machine cell formations for each period are summarized Table 5.21 and 5.22, respectively.

Table 5.21. Part Family Formations

		Part Families	
		Part Type	Total
Period 1	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,8,9	5
Period 2	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,6,8,9,10	7
Period 3	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,6,8,9,10	7

Table 5.22. Final Machine Cell Formations

		Machine Type							
		1	2	3	4	5	6	7	Total
Period 1	Cell 1	0	1	0	1	1	1	1	5
	Cell 2	1	1	1	2	0	0	0	5
Period 2	Cell 1	0	0	0	1	1	1	1	4
	Cell 2	1	1	1	2	0	0	0	5
Period 3	Cell 1	0	0	0	1	1	1	1	4
	Cell 2	1	1	1	1	0	0	0	4

The results indicate that Model III achieved a balanced machining capacity distribution throughout the planning period by grouping the machines into machine cells such that the variation in the number of machines from each type is minimized among periods. The required total machining capacity for machine type 2 decreased from 2 units to 1 unit in period 2, because of the dramatic decrease in demand of part types 6 and 10 in period 2. In order to minimize the total cost and achieve a balanced machining capacity distribution, the model removed one unit of machine type 2 from cell 1 at the end of period 1. The cost of this action is the sum of relocation cost incurred from removing machine type 2 from machine cell 1 and the sum of intercellular movement cost created by the removal of machine type 2 from machine cell 1. The savings of this action are the sum of extra unnecessary investment cost and idle time cost of machine type 2. Since the sum of investment cost and idle time cost of machine type 2 for one period were higher than the relocation and intercellular movement costs, the system removed machine type 2 to minimize the overall system cost. The same type of reasoning is valid for the removal of one unit of machine type 4 from machine cell 2, at the end of period 2. From the results, it is also seen that the model achieved a uniform part family composition throughout the planning horizon. Except part types 1, 6 and 10, which are exceptional part types, part family I contains part types 2,3 and 7, and part family II contains part types 4,5,8 and 9, for each period in the planning horizon. Maintaining the same part family compositions throughout the planning horizon,

makes the control of the manufacturing cells much easier and improves the efficiency. The overall optimal cost values of the final solution for the planning horizon, are given in Table 5.23.

Table 5.23. Final Optimal Cost Values

Cost Function	Cost Value($)
Total Capital Investment Cost	578,000
Total Idle Time Cost	49,971
Total Intercellular Movement Cost	14,000
Total Relocation Cost	10,000
Total Cost	**651,971**

5.4. Model IV, Multi Period

In this section, Model IV is tested by using the same example used to test Model III. The cost parameters used in the example, in addition to the ones used for testing Model III, are given in Table 5.24. The objective is to simultaneously develop the part family and machine cell configurations for each period in the planning horizon while attempting to minimize the exceptional elements and obtain a balanced machining capacity distribution throughout the planning horizon. Cost trade-offs among intercellular movement, machine duplication, subcontracting and machine relocation are under consideration. The list of the LINDO input files are in appendix III, and the resulting part family and machine cell formations are summarized in Table 5.25 and

5.26, respectively.

Table 5.24. Unit Subcontracting Costs

Part Type	Subcontracting Cost ($/unit)	Part Type	Subcontracting Cost ($/unit)
1	0.45	6	4.5
2	4.0	7	4.4
3	4.0	8	3.6
4	3.5	9	4.0
5	3.3	10	4.2

Table 5.25. Final Part Family Formations

		Part Families	
		Part Type	Total
Period 1	Cell 1	2,3,6,7,10	5
	Cell 2	4,5,8,9	4
Period 2	Cell 1	2,3,4,5,6,7,10	7
	Cell 2	4,5,8,9	4
Period 3	Cell 1	2,3,4,5,6,7,10	7
	Cell 2	4,5,8,9	4

Table 5.26. Final Machine Cell Formations

		Machine Type							
		1	2	3	4	5	6	7	Total
Period 1	Cell 1	0	1	0	1	1	1	1	5
	Cell 2	1	1	1	2	0	0	0	5
Period 2	Cell 1	0	1	0	1	1	1	1	5
	Cell 2	1	0	1	2	0	0	0	4
Period 3	Cell 1	0	1	0	1	1	1	1	5
	Cell 2	1	0	1	1	0	0	0	3

The results indicate that Model IV achieved a balanced machining capacity distribution throughout the planning horizon by grouping the machines into machine cells such that the variation in the number of machines from each type is minimized among periods. In order to minimize the total cost and achieve a balanced machining capacity distribution, one unit of machine type 2 and one unit of machine type 4 is removed from machine cell 2 at the end of the first and second period, respectively based on the same reasoning explained in section 5.3 for Model III. The final part family composition obtained for Model IV is different than the part family composition obtained for Model III since part type 1 is subcontracted for all the periods in the planning horizon. The model has subcontracted the part type 1 based on cost trade-offs since for each period the cost of producing part type 1 in the system is higher than the

cost of subcontracting part type 1. Furthermore, it is also seen that a uniform part family composition has been achieved throughout the planning horizon, since part family I contains part types 2,3,6,7 and 10, and part family II contains part types 8 and 9, for each period. The overall optimal cost values of the final solution are given in Table 5.27.

Table 5.27. Final Optimal Cost Values

Cost Function	Cost Value($)
Total Capital Investment Cost	578,000
Total Idle Time Cost	73,431
Total Intercelluiar Movement Cost	8,000
Total Machining Cost	386,000
Total Relocation Cost	10,000
Total Subcontracting Cost	28,800
Total Cost	**1,084,231**

5.5. Strategy Evaluation

In this section, a different type of multi period planning strategy, strategy I, is tested and the results are compared with the results of Model III and Model IV. The planning strategy under consideration, as contrary to the planning strategy developed in Model III and Model IV that takes into account the changes in part demand for all periods in the planning horizon, considers only the demand data in the first period

without recognizing the possibility of fluctuations in part demand and changes in machine cell configurations in future periods in the production planning horizon. Strategy I forms the part families and machine cells by taking into account the demand data in period I and pushes the system through the planning horizon without any modifications in the configuration of the machine cells. Throughout the planning horizon, machine cell configurations established in period I will be kept the same but composition of the part families might change from period to period, depending on the capacity requirements in each period.

During periods when there is excess machining capacity, strategy I will result in flexible systems with high capital investment and unutilized capacity costs. During periods when there is machining capacity shortage, strategy I will result in inflexible systems full of unsatisfied orders.

5.5.1. Comparison of Strategy I with Model III

In this section, strategy I is tested by the example used in section 5.3. Under strategy I, machine relocation cost is not considered since machine cell configurations will be kept the same throughout the planning horizon. Model I is used to form the part families and machine cells in period I, and for other periods, operations of the parts are assigned to the machine cells by using Model I with small modifications. In the

modified version of Model I, the values $Y_{k,j,t}$ will be assigned in advance and there will not be any cell size limitation constraints since the machine cell configurations will be kept the same throughout the planning horizon. The resulting part family formation under strategy I is given in Table 5.28. In Table 5.29, only the machine cell configuration in Period I is given since the machine cell configuration is kept the same after the first period under strategy I.

Table 5.28. Part Family Formation Under Strategy I

		Part Families	
		Part Type	Total
Period 1	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,8,9	5
Period 2	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,8,9	5
Period 3	Cell 1	1,2,3,6,7,10	6
	Cell 2	1,4,5,8,9	5

Table 5.29. Machine Cell Configuration Under Strategy I

Period 1	Machine Type							
	1	2	3	4	5	6	7	Total
Cell 1	0	1	0	1	1	1	1	5
Cell 2	1	1	1	2	0	0	0	5

The results indicate that under strategy I, the same part family composition is maintained throughout the planning horizon. Comparison of the cost functions for strategy I and Model III is given in Table 5.30.

Table 5.30 Comparison of the Cost Functions for Strategy I and Model III

Cost Functions	Period I		Period II		Period III	
	Strategy I	Model III	Strategy I	Model III	Strategy I	Model III
Machine Investment Cost	214,000	214,000	214,000	194,000	214,000	170,000
Idle Time Cost	15,169	15,169	28,127	16,127	36,675	18,675
Intercellular Movement Cost	2,000	2,000	2,000	6,000	2,000	6,000
Relocation Cost	0	5,000	0	5,000	0	0
Total Cost	231,169	236,169	244,127	221,127	252,675	194,675

When the resulting cost values are compared, it is seen that overall Model III has lower cost values than strategy I. Model III has lower capital investment cost in periods II and III, since it relocated the machines in the machine cells in order to achieve a balanced capacity distribution. Because of the fluctuations in part demand, the capacity requirement for total number of machines from type 2 and total number of machines from type 4 is reduced by one unit in periods II and III, respectively. Since Model III recognizes the fluctuations in part demand, it removed one unit of machine type 2 at the end of period I and one unit of machine type 4 at the end of period II from the system to reduce the capital investment cost, whereas strategy I kept the number of machine

type 2 and machine type 4 fixed throughout the planning horizon and resulted in higher capital investment cost, since the objective of strategy I was to push the system throughout the planning horizon, which was created in period I, without recognizing the fluctuations in part demand in periods II and III.

Keeping the extra unit of machine type 2 in the system for periods II and III resulted in a flexible system, which caused the strategy I to have lower intercellular movement cost. Under Model III, after period I, part types 6 and 10 are transported to the machine cell 2 to be processed on machine type 2, since machine type 2 has been removed from machine cell I at the end of period I in order to reduce the capital investment cost. However, keeping the extra machines in the system lowered the machine cell utilizations, thus increasing the total idle time cost of strategy I by outweighing the savings obtained from the reductions in intercellular movement cost. The resulting machine cell utilizations under Strategy I and Model III are given in Table 5.31. The results show that consideration of the fluctuations in part demand in future production periods during the planning stage results in manufacturing systems with lower cost values and higher resource utilizations. The graphical representation of the resulting cost values are given in Figure 5.1.

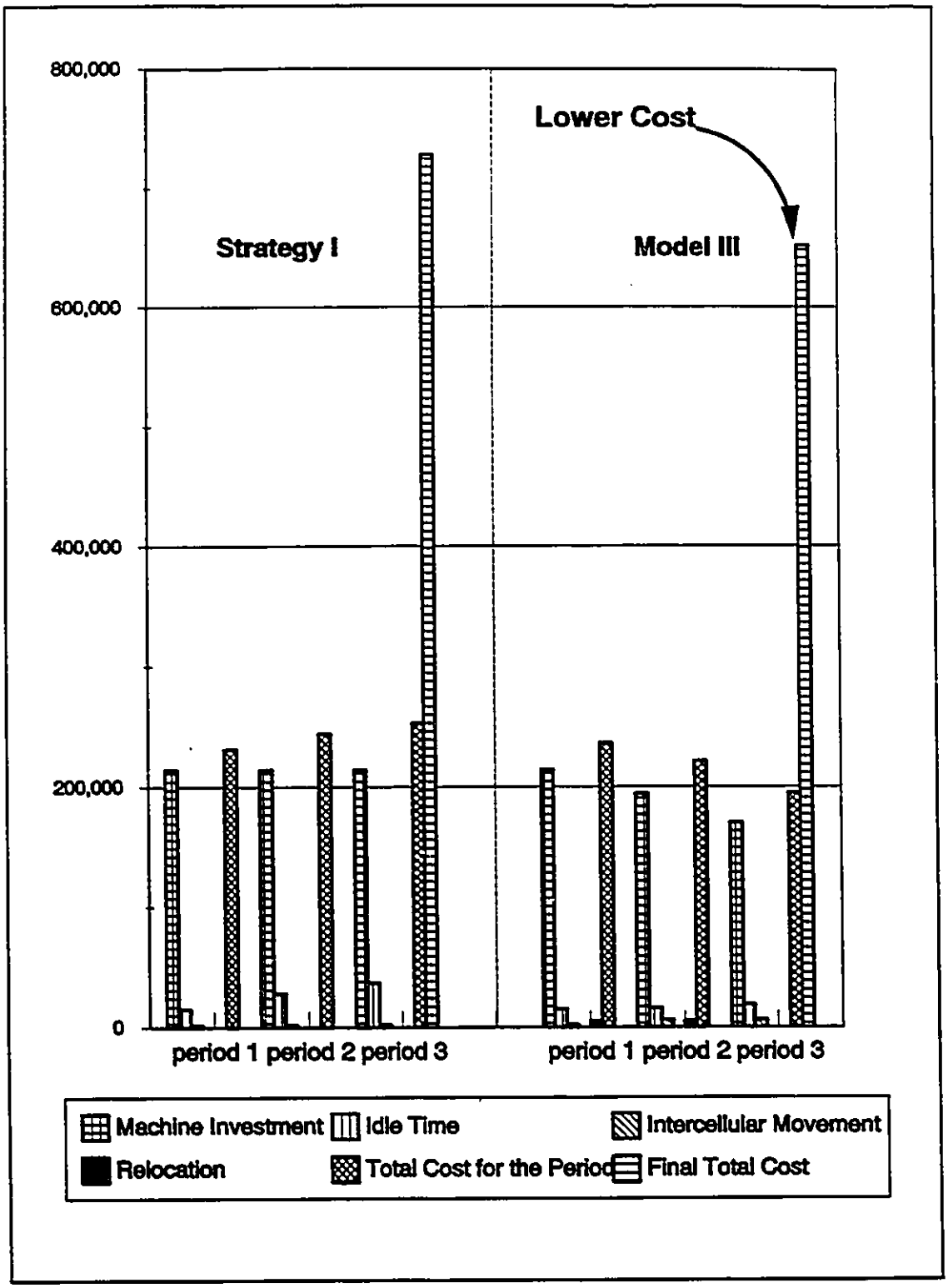

Figure 5.1. Comparison of the Cost Functions Under Strategy I and Model III

Table 5.31. Machine Cell Utilizations Under Strategy I and Model III

		STRATEGY I		MODEL III	
		Cell 1	Cell 2	Cell 1	Cell 2
Period I	Total Machining Capacity in the Cell	10,000	10,000	10,000	10,000
	Total Processing Time Within the Cell	9,493	6,742	9,493	6,742
	Total Idle Time	507	3,258	507	3,258
	Cell Utilization	0.95	0.675	0.95	0.675
	System Utilization	0.812		0.812	
Period II	Total Machining Capacity in the Cell	10,000	10,000	8,000	10,000
	Total Processing Time within the Cell	6,362	7,148	5,582	7,928
	Total Idle Time	3,638	2,852	2,418	2,072
	Cell Utilization	0.64	0.72	0.70	0.793
	System Utilization	0.68		0.75	
Period III	Total Machining Capacity in the Cell	10,000	10,000	8,000	8,000
	Total Processing Time within the Cell	5,016	6,366	4,451	6,931
	Total Idle Time	4,984	3,634	3,549	1,069
	Cell Utilization	0.5	0.64	0.556	0.866
	System Utilization	0.57		0.711	

5.5.2. Comparison of Strategy I with Model IV

In this section, strategy I is tested by the example used in section 5.4. Model II is used to form the part families and machine cells in period I, and for other periods part family compositions are obtained by using Model II with small modifications. In the modified version of Model II, the values of $Y_{k,j,t}$ will be assigned in advance and there will not be any cell size limitation constraints since machine cell configurations will be kept the same throughout the planning horizon. The resulting part family formation is given in Table 5.32. In Table 5.33, only the machine cell configuration in Period I is given, since the machine cell configuration is kept the same after the first period under strategy I.

Table 5.32. Part Family Formation Under Strategy I

		Part Families	
		Part Type	Total
Period 1	Cell 1	2,3,6,7,10	5
	Cell 2	4,5,8,9	4
Period 2	Cell 1	2,3,6,7,10	5
	Cell 2	4,5,8,9	4
Period 3	Cell 1	2,3,6,7,10	5
	Cell 2	4,5,8,9	4

Table 5.33. Machine Cell Configuration Under Strategy I

Period 1	Machine Type							
	1	2	3	4	5	6	7	Total
Cell 1	0	1	0	1	1	1	1	5
Cell 2	1	1	1	2	0	0	0	5

The results indicate that under strategy I, the same part family composition is maintained throughout the planning horizon. Part type 1 has been subcontracted for all the periods under consideration based on economic cost trade-offs. Comparison of the cost functions for strategy I and Model IV is given in Table 5.34.

Table 5.34. Comparison of the Cost Functions for Strategy I and Model IV

Cost Functions	Period I		Period II		Period III	
	Strategy I	Model IV	Strategy I	Model IV	Strategy I	Model IV
Machine Investment Cost	214,000	214,000	214,000	194,000	214,000	170,000
Idle Time Cost	22,869	22,869	36,187	24,187	44,375	26,375
Intercellular Movement Cost	0	0	0	4,000	0	4,000
Machining Cost	156,598	156,598	125,961	125,961	103,441	103,441
Relocation Cost	0	5,000	0	5,000	0	0
Subcontracting Cost	9,450	9,450	9,900	9,900	9,450	9,450
Total	402,917	407,917	386,048	363,048	371,266	313,266

When the resulting cost values are compared, overall it is seen that Model IV has lower cost values than strategy I. Model IV has lower capital investment costs in periods II and III since it removed one unit of machine type 2 at the end of period I and one unit of machine type 4 at the end of period II from the system, whereas strategy I kept the machine cell configuration the same throughout the planning horizon, based on the same reasoning discussed in section 5.5.1. Keeping the extra unit of machine type 2 in the system for periods II and III resulted in a flexible system containing two independent machine cells, which caused the strategy I to have zero intercellular movement cost. Under Model IV, after period I, part types 4 and 5 are transported to the machine cell 1 to be processed on machine type 2, since machine type 2 has been removed from machine cell II at the end of period I to reduce the capital investment cost. However, keeping the extra machines in the system lowered the machine cell utilizations, thus increasing the total idle time cost of strategy I by outweighing the saving obtained from zero intercellular movement cost. The resulting machine cell utilizations under Strategy I and Model IV are given in Table 5.35. The total machining cost for strategy I and Model IV are the same, since each operation of a part type can only be processed on a single type of machine. Once again it has been shown that consideration of fluctuations in future part demand resulted in a manufacturing system with lower costs and higher resource utilizations. The graphical representation of the resulting cost values is given in Figure 5.2.

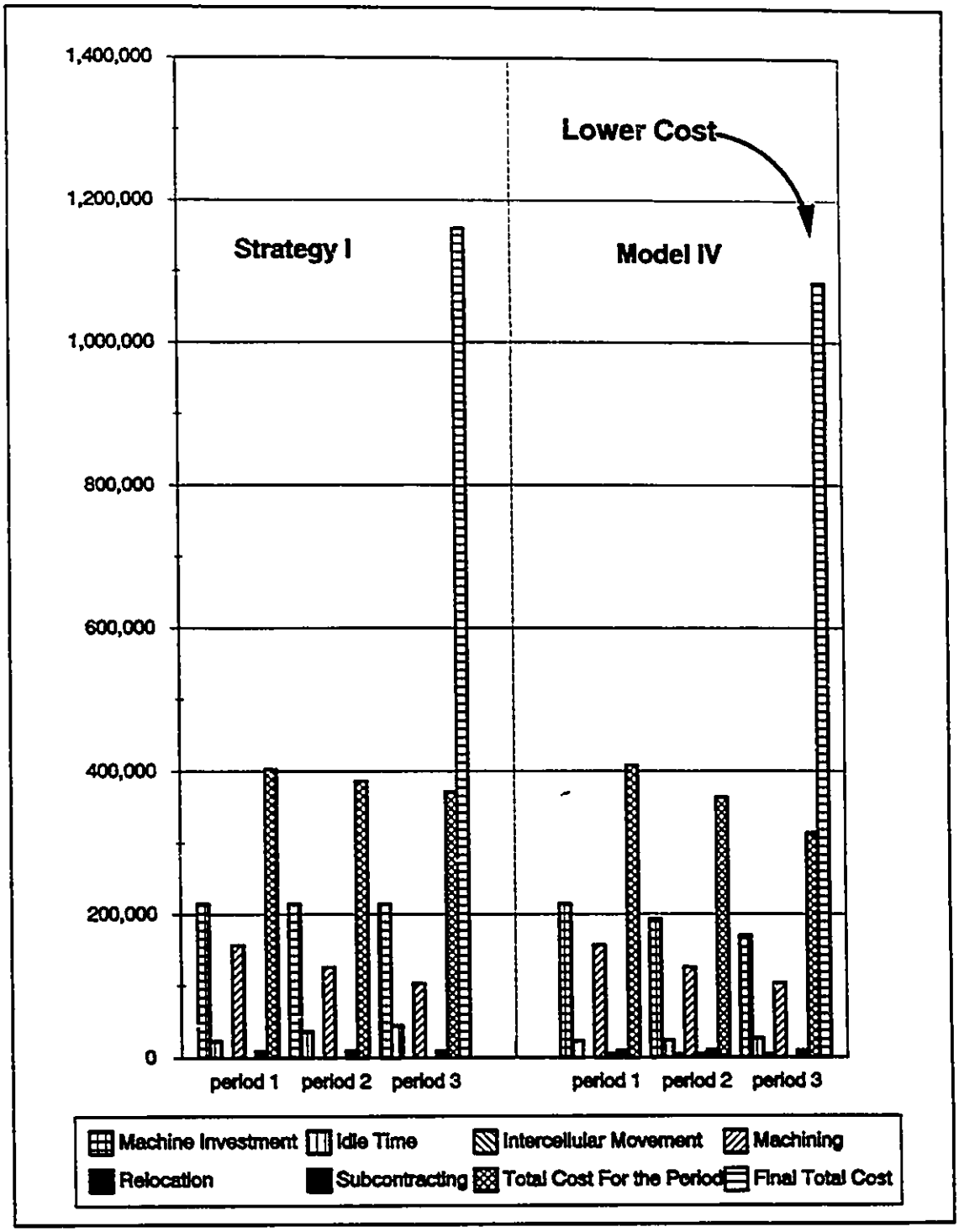

Figure 5.2. Comparison of the Cost Functions Under Strategy I and Model IV

Table 5.35. Machine Cell Utilizations Under Strategy I and Model IV

		STRATEGY I		MODEL IV	
		Cell 1	Cell 2	Cell 1	Cell 2
Period I	Total Machining Capacity in the Cell	10,000	10,000	10,000	10,000
	Total Processing Time Within the Cell	8,933	5,377	8,933	5,377
	Total Idle Time	1,067	4,623	1,067	4,623
	Cell Utilization	0.89	0.54	0.89	0.54
	System Utilization	0.715		0.715	
Period II	Total Machining Capacity in the Cell	10,000	10,000	10,000	8,000
	Total Processing Time within the Cell	5,776	5,719	6,981	4,514
	Total Idle Time	4,224	4,281	3,019	3,486
	Cell Utilization	0.578	0.572	0.70	0.56
	System Utilization	0.575		0.639	
Period III	Total Machining Capacity in the Cell	10,000	10,000	10,000	6,000
	Total Processing Time within the Cell	4,456	5,001	5,471	3,986
	Total Idle Time	5,544	4,999	4,529	2,014
	Cell Utilization	0.45	0.5	0.55	0.66
	System Utilization	0.473		0.591	

5.6. Evaluation of Genetic Algorithms for Single Period Models

In this section, applicability and efficiency of genetic algorithms, GENMOD I and GENMOD II, are tested by using a large size example. The results of genetic algorithms are compared with the results of Model I and Model II in terms of computational time, resulting cost functions machine cell utilizations.

The example used to test the genetic algorithms includes 20 parts and 12 machine types. During the single period planning horizon, 2 machine cell formation is considered, 2000 hours of machine capacity is assigned to each of the machines and for each intercellular movement occurred in the system, a cost of $2000 is charged. Various cost parameters used during the calculations are given in Table 5.36. Machine requirements and annual demand of the parts in the system are given in Table 5.37.

Table 5.36. Capital, Machining and Idle Time Costs

Machine Type	Capital Cost ($/year)	Machining Cost ($/hr)	Idle Time Cost ($/hr)
1	25,000	13	2
2	26,000	10	4
3	23,000	11	5
4	15,000	13	2
5	15,000	12	5
6	24,000	11	4
7	20,000	10	3
8	18,000	9	4
9	17,000	10	5

Machine Type	Capital Cost ($/year)	Machining Cost ($/hr)	Idle Time Cost ($/hr)
10	16,000	11	4
11	22,000	10	3
12	21,000	9	3

Table 5.37. Machine Requirements and Annual Demand of the Part Types

Part Type	Machine Type												Annual Demand
	1	2	3	4	5	6	7	8	9	10	11	12	
1	0	0	1.9	0	0	1.8	0	0	0	0	0	0	16,000
2	0	2.3	0	0	0	2.8	0	0	0	0	0	0	18,000
3	0	2.2	0	0	2.2	0	0	0	1.9	0	0	0	17,000
4	0	2.7	0	0	0	0	3.1	0	0	2.1	0	0	17,000
5	0	0	1.7	0	1.6	0	0	0	0	1.8	0	0	17,000
6	2.9	0	0	0	1.8	0	0	0	2.5	0	0	0	19,000
7	1.6	0	0	1.7	0	0	1.4	0	0	0	1.8	0	15,000
8	0	0	1.8	0	0	0	2.9	0	0	0	0	2.8	16,000
9	0	1.7	0	0	0	0	2.8	0	0	2.9	0	0	16,000
10	1.5	0	0	1.8	0	0	0	0	0	1.8	0	0	19,000
11	0	0	1.5	0	1.7	0	0	0	2.1	0	0	0	16,000
12	2.0	0	0	1.6	0	0	0	2.4	0	0	0	0	18,000
13	0	0	2.1	0	0	2.3	0	0	0	0	0	0	14,000
14	0	2.5	0	1.9	0	0	0	2.0	0	0	0	0	17,000
15	0	0	2.5	0	0	0	3.1	0	0	0	0	3.2	16,000
16	2.3	0	0	0	2.8	0	0	0	0	2.9	0	0	20,000
17	0	0	2.9	2.1	0	0	0	0	0	0	2.7	0	16,000
18	1.3	0	0	1.7	0	0	0	0	0	0	0	0	14,000
19	1.7	0	0	2.0	0	3.0	0	0	0	0	0	0	18,000
20	0	2.1	0	0	0	3.2	0	0	0	1.6	0	0	19,000

5.6.1. Evaluation of GENMOD I

The example given in section 5.6. is used as an input to both Model I and GENMOD I. The resulting part family and machine cell formations obtained from Model I are given in Tables 5.38 and 5.39, respectively. To test GENMOD I, 100000 generations are created, of which each generation includes a population of size 60. During the calculations, the probability of crossover and probability of mutation is taken as 0.89 and 0.089, respectively. The resulting part family and machine cell formations obtained from GENMOD I are given in Tables 5.40 and 5.41, respectively. The comparison of the cost functions for Model I and GENMOD I are given in Table 5.42.

Table 5.38. Part Family Formation Under Model I

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,4,7,9,10,12,13,14,17,**18**,19	11
Cell 2	2,3,5,6,8,11,15,16,**18**,20	10

Table 5.39. Final Machining Cell Composition Under Model I

Machining Cell	Machine Type												Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	11	12	
Cell 1	1	1	1	2	0	1	1	1	0	1	1	0	10
Cell 2	1	1	1	0	2	1	1	0	1	1	0	1	10

Table 5.40. Final Part Family Composition Under GENMOD I

Machining Cells	Part Families	
	Part Type	Total
Cell 1	3,5,6,8,10,11,12,14,15,**20**	10
Cell 2	1,2,4,5,7,9,**10**,13,16,17,18,19,20	13

Table 5.41. Final Machining Cell Composition Under GENMOD I

Machining Cell	Machine Type												Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	11	12	
Cell 1	1	1	1	1	1	0	1	1	1	0	0	1	9
Cell 2	1	1	1	1	1	2	1	0	0	2	1	0	11

Table 5.42. Comparison of the Cost Functions for GENMOD I and Model I

Cost Functions	**GENMOD I**	**Model I**
Machine Investment Cost	406,000	406,000
Idle Time Cost	17,262	17,262
Intercellular Movement Cost	6,000	2,000
Total Cost	**429,262**	**425,262**

When the final results are compared, it is seen that part family and machine cell formations obtained under GENMOD I and Model I are different from each other.

GENMOD I formed two part families with 3 exceptional part types which are: 5,10 and 20. Machine cell I is formed of 9 machines, where as, machine cell II is formed of 11 machines. Model I grouped the part types into two part families with only one exceptional part type which is part type 18 and each machine cell contained 10 machines. Under GENMOD I and Model I, the total number of machines from each machine type in the system is the same.

GENMOD I found a sub-optimal total cost value with an error of 0.49%. The graphical illustration of the average population cost value during generations, is given in Figure 5.3. The decreasing slope of the average cost curve is an indication of GENMOD I's capability to generate part family and machine cell formations with lower cost values at each generation. For the following two cost functions: Machine Investment Cost and Idle Time Cost; GENMOD I succeeded in obtaining the same results obtained under Model I. The machine investment cost and idle time cost is the same under both models since the total number of machines from each type in the system is the same. GENMOD I obtained higher intercellular movement costs as compared to Model I, since the part family formation obtained under GENMOD I included three exceptional part types, whereas the part family formation obtained under Model I included only one exceptional part type. Even though GENMOD I obtained optimal results for the first two cost functions because of the higher intercellular movement cost, the overall total cost function value under GENMOD I turned out to

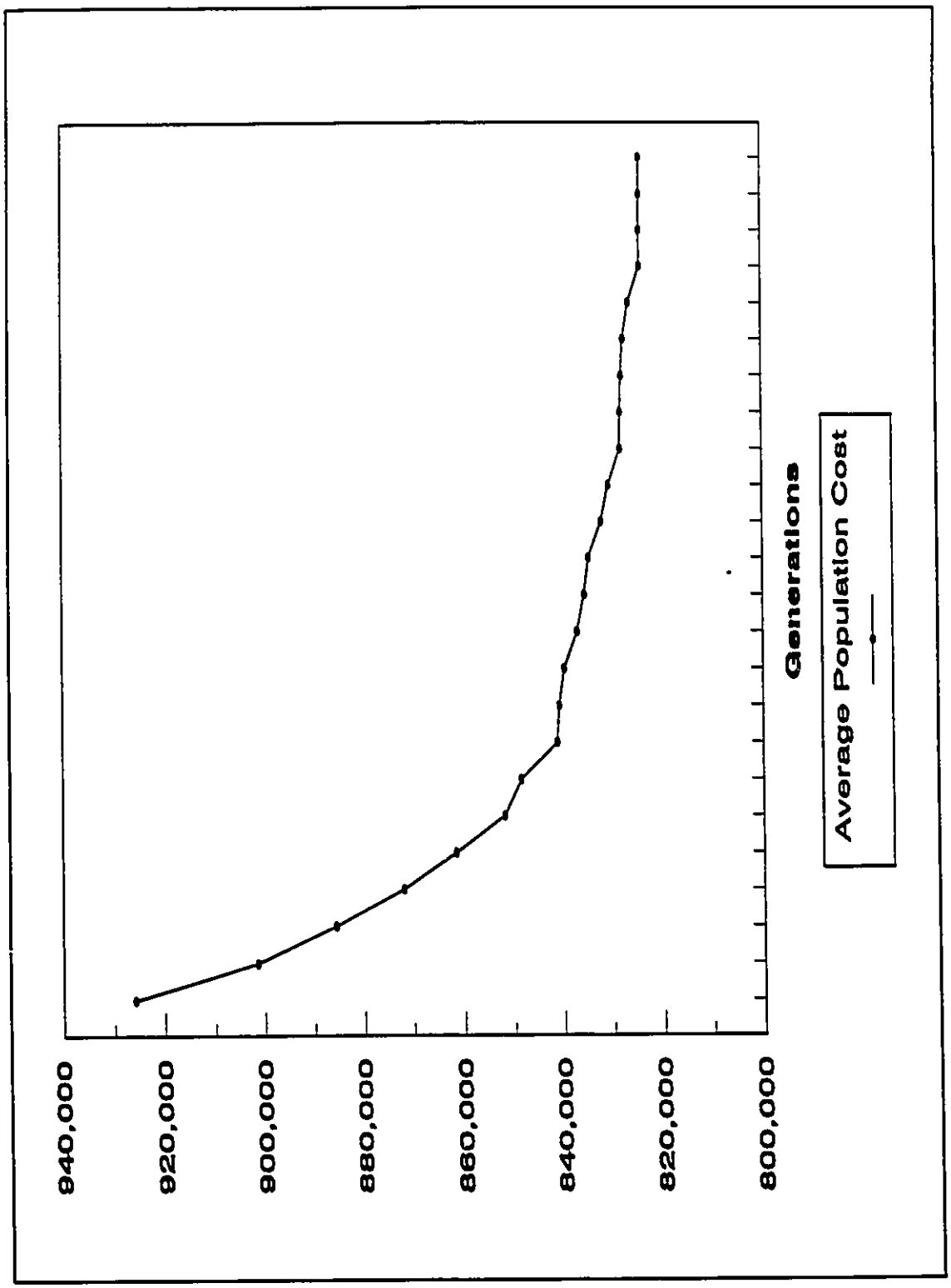

Figure 5.3. Average Population Cost Under GENMOD I

be sub-optimal.

The overall system utilization under Model I and GENMOD I is the same, since the total number of machines from each machine type is the same under both models However, the utilization of each machine cell shows slight variation under each model. The utilization of the machine cells under both of the models is given in Table 5.43.

Table 5.43. Machine Cell Utilizations Under GENMOD I and Model I

	GENMOD I		MODEL I	
	Cell 1	Cell 2	Cell 1	Cell 2
Total Machining Capacity in the Cell	18,000	22,000	20,000	20,000
Total Processing Time Within the Cell	15,596	19,774	17,450	17,920
Total Idle Time	2,404	2,226	2,550	2,080
Cell Utilization	0.87	0.90	0.873	0.896
System Utilization	0.88		0.88	

The computational time for Model I to reach the optimal solution was 6:12 hours, where as, the computational time for GENMOD I to reach the sub-optimal solution was 0:33 hours, *(91.1 % reduction in computational time).* Overall, the results obtained encourage the use of GENMOD I since it found a sub-optimal solution with low percentage error by obtaining a fairly acceptable part family and machine cell formation in less computational time as compared to Model I.

5.6.2. Evaluation of GENMOD II

The example given in section 5.6. is used to test Model II and GENMOD II. The resulting part family formation obtained from Model II is given in Table 5.44. The machine cell formation obtained under Model II is the same as obtained from ModelI, which was given in Table 5.39. In order to test GENMOD II, 100000 generations are created. Each generation includes a population of size 60. After 60000 generations subcontracting is applied. During the calculations, the probability of crossover and probability of mutation is taken as 0.89 and 0.089, respectively. The resulting part family and machine cell formations obtained under GENMOD II are given in Tables 5.45 and 5.46, respectively. Additionally, the unit subcontracting cost of each part type is given in Table 5.47. The comparison of the cost functions under Model II and GENMOD II, is given in Table 5.48.

Table 5.44. Final Part Family Composition under Model II

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,4,7,9,10,12,13,14,17,19	10
Cell 2	2,3,5,6,8,11,15,16,20	9

Table 5.45. Final Part Family Composition under GENMOD II

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,2,3,4,6,8,9,10,11,12,13,14,20	13
Cell 2	5,7,8,15,16,17,19,20	8

Table 5.46. Final Machining Cell Composition under GENMOD II

Machining Cell	Machine Type												Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	11	12	
Cell 1	1	2	1	1	1	1	1	1	1	1	0	0	11
Cell 2	1	0	1	1	1	1	1	0	0	1	1	1	9

Table 5.47. Unit Subcontracting Costs

Part Type	Cost ($/unit)	Part Type	Cost ($/unit)	Part Type	Cost ($/unit)	Part Type	Cost ($/unit)
1	4.6	6	4.9	11	4.5	16	5.8
2	4.8	7	5.8	12	5.2	17	5.4
3	5.1	8	5.3	13	4.3	18	0.5
4	4.8	9	4.9	14	5	19	5.9
5	4.5	10	5.6	15	5.4	20	5.2

Table 5.48. Comparison of the Cost Functions Under GENMOD II and Model II

Cost Functions	GENMOD II	Model II
Machine Investment Cost	406,000	406,000
Idle Time Cost	18,660	18,660
Machining Cost	381,876	381,876
Intercellular Movement Cost	4,000	0
Subcontracting Cost	7,000	7,000
Total Cost	817,536	813,536

The resulting part family and machine cell formations obtained under GENMOD II and Model II, are different from each other. Part type 18 is

subcontracted in both of the models based on economic considerations. Model II grouped the part types into two part families without creating any exceptional part. Machine cells I and II are independent, i.e. there is not any part type moving among the machine cells. GENMOD II resulted in two part families with two exceptional part types which are: 8 and 20. Machine cell I contains 11 machines and machine cell II contains 9 machines. The total number of machines from each type in the system is the same under both of the models.

GENMOD II found a sub-optimal total cost value with an error of 0.49 %. The graphical illustration of the average population cost values during generations is given in Figure 5.4. The decreasing slope of the average cost curve indicates GENMOD II's capability to generate part family and machine cell formations with lower cost values at each generation. For the following four cost functions: Machine Investment Cost, Idle Time Cost, Machining Cost and Subcontracting Cost; GENMOD II obtained the same results as in Model II. The machine investment and idle time cost is the same under both of the models since the total number of machines from each type in the system is the same. 0.49 % error in the final solution obtained under GENMOD II comes from higher intercellular movement cost. GENMOD II has higher intercellular movement costs as compared to Model II because of the existence of exceptional part types. Even though, under GENMOD II the results for the first four cost functions are optimal, higher intercellular movement cost makes the overall total cost function value

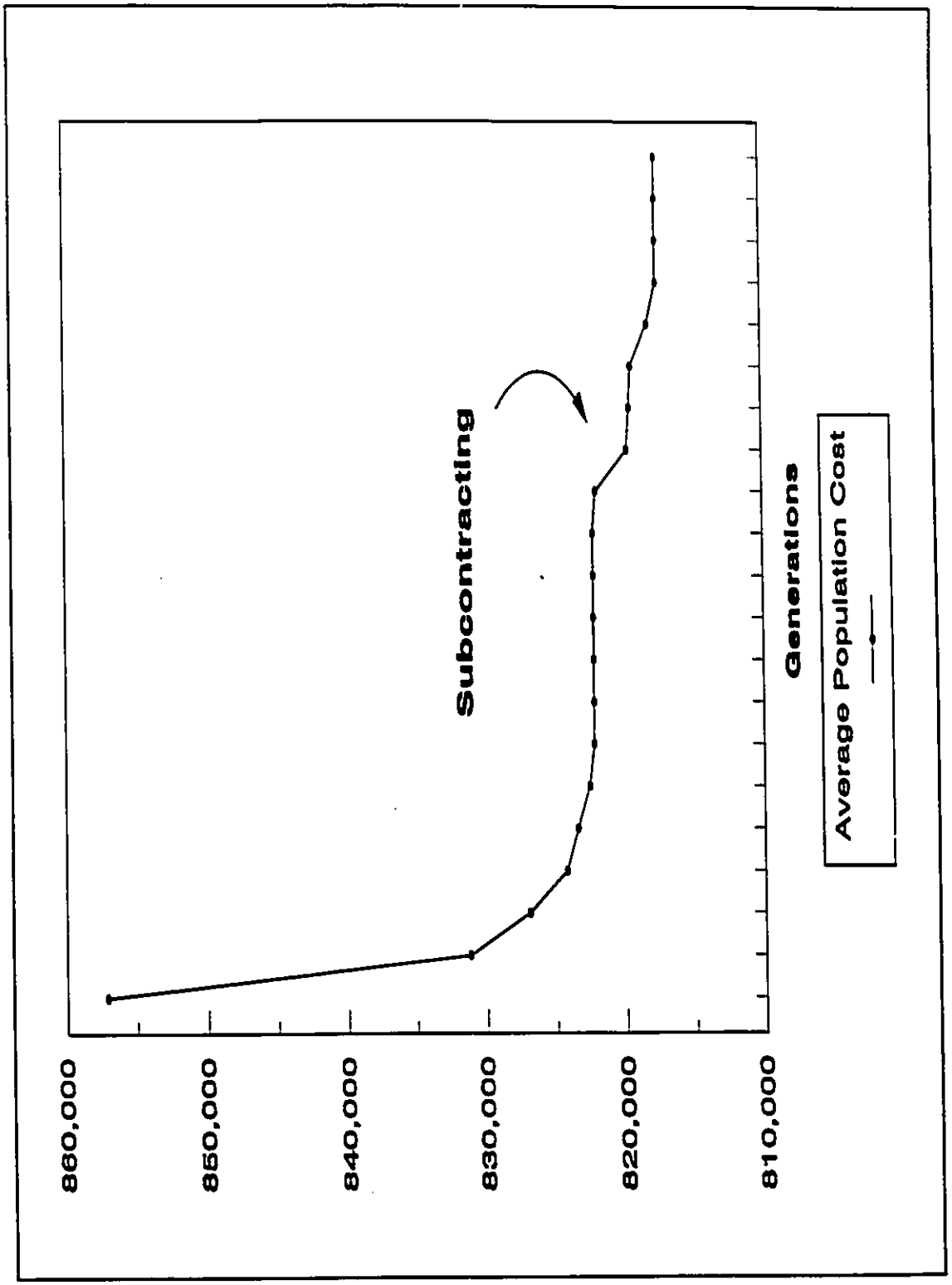

Figure 5.4. Average Population Cost Under GENMOD II

under GENMOD II to be sub-optimal.

The overall system utilization is the same under both models, since the total number of machines from each type in the system is the same. However, the utilization of each machine cell shows slight variations under each model. The utilization of machine cells under both of the models is given in Table 5.49.

Table 5.49. Machine Utilization Under GENMOD II and Model II

	GENMOD II		MODEL II	
	Cell 1	Cell 2	Cell 1	Cell 2
Total Machining Capacity in the Cell	22,000	18,000	20,000	20,000
Total Processing Time Within the Cell	19,518	15,153	17,617	17,054
Total Idle Time	2,482	2,847	2,383	2,946
Cell Utilization	0.887	0.84	0.88	0.853
System Utilization	0.86		0.86	

The computational time for Model II to reach the optimal solution was 13:18 hours, whereas the computational time for GENMOD II to reach the sub-optimal solution was 0:33 hours, *(95.7 % reduction in computational time)*. Overall, the results obtained indicate that GENMOD II found a sub-optimal solution with low percentage error by obtaining a fairly acceptable part family and machine cell formation in less computational time as compared to Model II.

5.7. Evaluation of Genetic Algorithms for Multi Period Models

In this section, efficiency and applicability of genetic algorithms GENMOD III and GENMOD IV, are tested by using the example developed in section 5.3. The length of the planning horizon, machine requirements of the parts for each period and various cost parameters are taken the same, as specified in section 5.3.

5.7.1. Evaluation of GENMOD III

The example given in section 5.3. is used to test GENMOD III. During the calculations, 90000 generations are created, of which each generation included a population of size 60. The probability of crossover and probability of mutation is taken as 0.9989 and 0.04955, respectively. The resulting part family and machine cell formations for each period obtained under GENMOD III are given in Tables 5.50 and 5.51, respectively. The comparison of the cost functions under GENMOD III and Model III is given in Table 5.52.

Table 5.50. Part Family Formation Under GENMOD III

		Part Families	
		Part Type	Total
Period 1	Cell 1	1,2,5,6,7,8,9,10	8
	Cell 2	1,2,3,4,6,10	6
Period 2	Cell 1	1,2,3,6,7,8,9,10	8
	Cell 2	1,4,5,6,10	5
Period 3	Cell 1	1,2,3,6,7,8,9,10	8
	Cell 2	1,4,5,6,10	5

Table 5.51. Machine Cell Formation Under GENMOD III

		Machine Type							
		1	2	3	4	5	6	7	Total
Period 1	Cell 1	1	1	1	2	0	1	0	6
	Cell 2	0	1	0	1	1	0	1	4
Period 2	Cell 1	1	0	1	2	0	1	1	6
	Cell 2	0	1	0	1	1	0	0	3
Period 3	Cell 1	1	0	1	1	0	1	1	5
	Cell 2	0	1	0	1	1	0	0	3

Table 5.52. Cost Comparison for GENMOD III and Model III

Cost Functions	GENMOD III	Model III
Machine Investment Cost	578,000	578,000
Idle Time Cost	49,971	49,971
Intercellular Movement Cost	20,000	14,000
Relocation Cost	20,000	10,000
Total Cost	**667,971**	**651,971**

The part family formations obtained under GENMOD III and Model III are the same in all periods in terms of exceptional part types, except the first period. During period I, GENMOD III grouped the parts into two part families with four exceptional

part types which are: 1, 2, 6 and 10; where as, Model III formed the part families with only one exceptional part type which is part type 1.

The results indicate that GENMOD III achieved a balanced machining capacity distribution throughout the planning horizon by grouping the machines into machine cells in a such way that the variation in the number of machines from each machine type is minimized among periods. Under GENMOD III and Model III, one unit of machine type 2 and one unit of machine type 4 is removed from the system at the end of period I and period II, respectively because of the decrease in the demand of the part types that require these two machine types. In addition to the removal of machine types 2 and 4, GENMOD III removed machine type 7 from machine cell 2 and reinstalled in machine cell 1 at the end of period I which resulted in higher relocation costs for GENMOD III. Under Model III and GENMOD III, the total number of machines from each machine type in each period is the same.

The results in Table 5.52 indicate that the total cost value found under GENMOD III is sub-optimal with an error of 1.46 %. The graphical representation of the average population cost values during generations is given in Figure 5.5. The decreasing slope of the average cost curve is an indication of GENMOD III's capability to generate part family and machine cell formations with lower cost values at each generation. For the following two cost functions: Machine Investment Cost and Idle Time Cost; GENMOD III succeeded in obtaining the same results obtained under

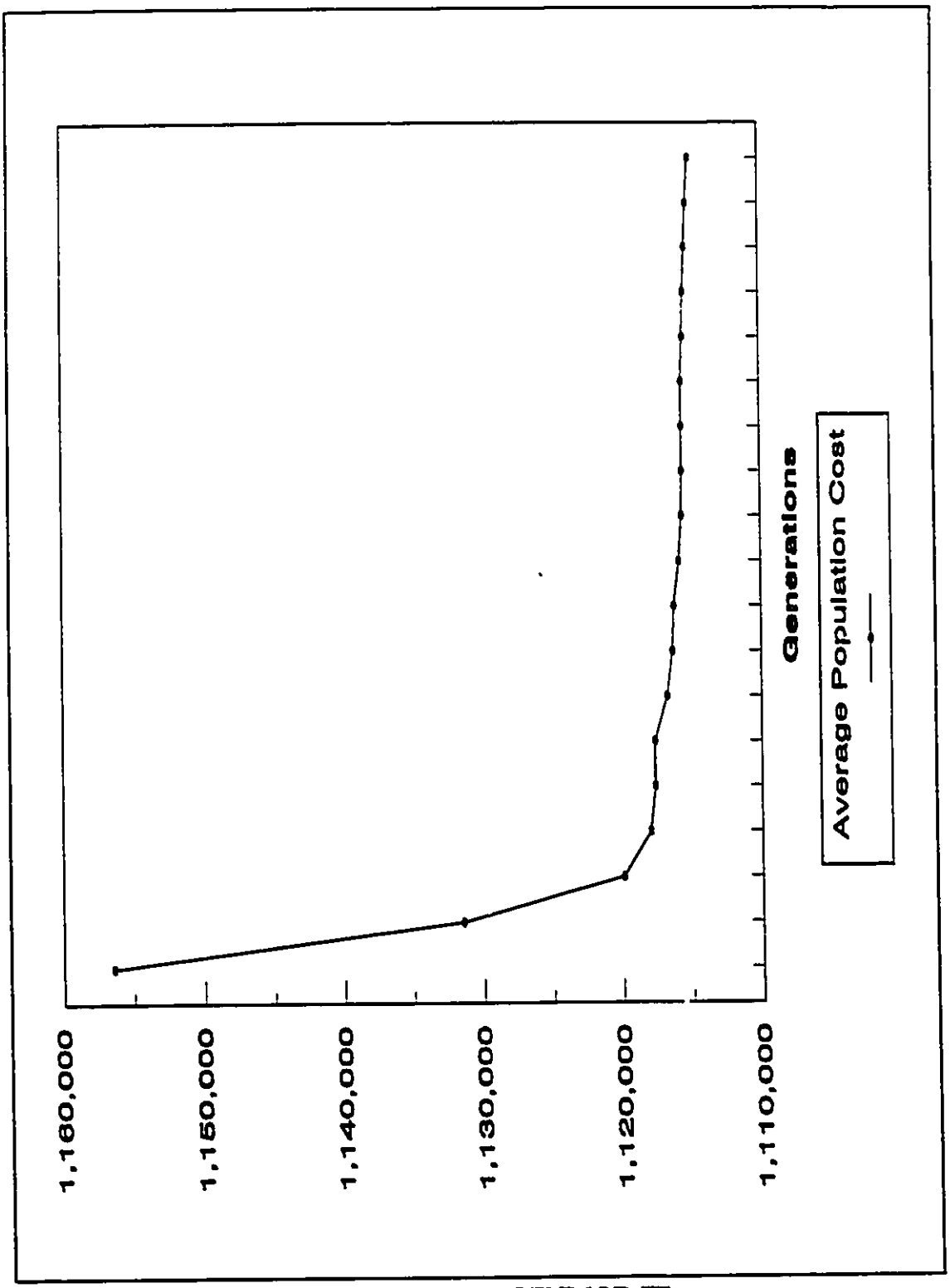

Figure 5.5. Average Population Cost Under GENMOD III

Model III. GENMOD III's intercellular movement and relocation costs are higher than Model III's. Existence of three more exceptional part types in period I resulted in higher intercellular movement cost for GENMOD III. The relocation cost is higher in GENMOD III because of the relocation of machine type 7 from machine cell 2 to machine cell 1, at the end of period I. Even though GENMOD III obtained optimal results for the first three cost functions because of the higher intercellular movement and relocation costs, the overall total cost function value under GENMOD III became sub-optimal.

The overall system utilization under both of the models is equal, since in each period the total number of machines from each type is the same. However, the utilization of each machine cell during periods shows variation under each model. The utilization of the machine cells under both of the models is given in Table 5.53.

The results given in Table 5.53 indicate that for each period during the planning horizon, GENMOD III achieved better machine cell utilizations than Model III, by assigning the machining capacities in a more balanced way. The computational time for Model III to reach the optimal solution was 4:54 hours, where as, the computational time for GENMOD III to reach the sub-optimal solution was 0:50 hours, *(83.2% reduction in computational time).*

Table 5.53. Machine Cell Utilizations Under GENMOD III and Model III

		GENMOD III		MODEL III	
		Cell 1	Cell 2	Cell 1	Cell 2
Period I	Total Machining Capacity in the Cell	12,000	8,000	10,000	10,000
	Total Processing Time Within the Cell	9,424	6,811	9,493	6,742
	Total Idle Time	2,576	1,189	507	3,258
	Cell Utilization	0.785	0.85	0.95	0.675
	System Utilization	0.812		0.812	
Period II	Total Machining Capacity in the Cell	12,000	6,000	8,000	10,000
	Total Processing Time within the Cell	8,736	4,774	5,582	7,928
	Total Idle Time	3,264	1,226	2,418	2,072
	Cell Utilization	0.728	0.795	0.70	0.793
	System Utilization	0.75		0.75	
Period III	Total Machining Capacity in the Cell	10,000	6,000	8,000	8,000
	Total Processing Time within the Cell	7,456	3,926	4,451	6,931
	Total Idle Time	2,544	2,074	3,549	1,069
	Cell Utilization	0.746	0.654	0.556	0.866
	System Utilization	0.711		0.711	

Even if the results under GENMOD III are sub-optimal, they encourage the use of GENMOD III because of its capability to achieve balanced machine utilizations and obtain an acceptable solution in less amount of time as compared to Model III.

5.7.2. Evaluation of GENMOD IV

The example given in section 5.3 is used to test GENMOD IV. In order to test GENMOD IV, 120000 generations are created, of which each generation included a population of size 60. The probability of crossover and probability of mutation is taken as 0.9989 and 0.0495, respectively. After 80000 generations, subcontracting is applied. The resulting part family and machine cell formations under GENMOD IV are given in Tables 5.54 and 5.55, respectively. The comparison of the cost function under Model IV and GENMOD IV is given in Table 5.56.

Table 5.54. Part Family Formation Under GENMOD IV

		Part Families	
		Part Type	Total
Period 1	Cell 1	2,3,4,5,6,7,10	7
	Cell 2	6,8,9,10	4
Period 2	Cell 1	2,3,4,5,6,7,10	7
	Cell 2	6,8,9,10	4
Period 3	Cell 1	2,3,4,5,6,7,10	7
	Cell 2	2,4,6,8,9,10	6

Table 5.55. Machine Cell Formation Under GENMOD IV

		Machine Type							
		1	2	3	4	5	6	7	Total
Period 1	Cell 1	0	1	0	2	0	1	1	5
	Cell 2	1	1	1	1	1	0	0	5
Period 2	Cell 1	0	1	0	2	0	1	1	5
	Cell 2	1	0	1	1	1	0	0	4
Period 3	Cell 1	0	1	0	1	0	1	1	4
	Cell 2	1	0	1	1	1	0	0	4

Table 5.56. Cost Comparison for GENMOD IV and Model IV

Cost Functions	GENMOD IV	Model IV
Machine Investment Cost	578,000	578,000
Idle Time Cost	73,431	73,431
Machining Cost	386,000	386,000
Intercellular Movement Cost	24,000	8,000
Relocation Cost	10,000	10,000
Subcontracting Cost	28,800	28,800
Total Cost	**1,100,231**	**1,084,231**

The part family formations obtained under GENMOD IV and Model IV are different in all periods, except period 2, in terms of the number exceptional part types. Under both models, part type 1 is subcontracted in all periods based on economic cost

considerations. During period 1, Model IV obtained two independent part families, whereas GENMOD IV formed two part families with two exceptional part types which are: 6 and 10. During period 3, Model IV resulted in part families with two exceptional part types, which are: 4 and 5, where as, GENMOD IV created four exceptional part types which are 2,4,6 and 10.

The results indicate that GENMOD IV achieved a balanced machining capacity distribution throughout the planning horizon by grouping the machines into machine cells such that the variation in the number of machines from each machine type is minimized among periods. Under GENMOD IV and Model IV, one unit of machine type 2 and one unit of machine type 4 is removed from the system at the end of periods I and II, respectively because of the decrease in the demand for the capacity requirements of these two machines. Under Model IV and GENMOD IV, the total number of machines from each machine type in each period is the same.

The results in Table 5.56 indicate that the total cost value found under GENMOD IV is sub-optimal with an error of 1.47 %. The graphical representation of the average population cost values during generations is given in Figure 5.6. The decreasing slope of the average cost curve is an indication of GENMOD IV's capability to generate part family and machine cell formations with lower cost values at each generation. For all the cost functions, except the higher total intercellular movement cost function, GENMOD IV achieved the same results obtained under Model IV.

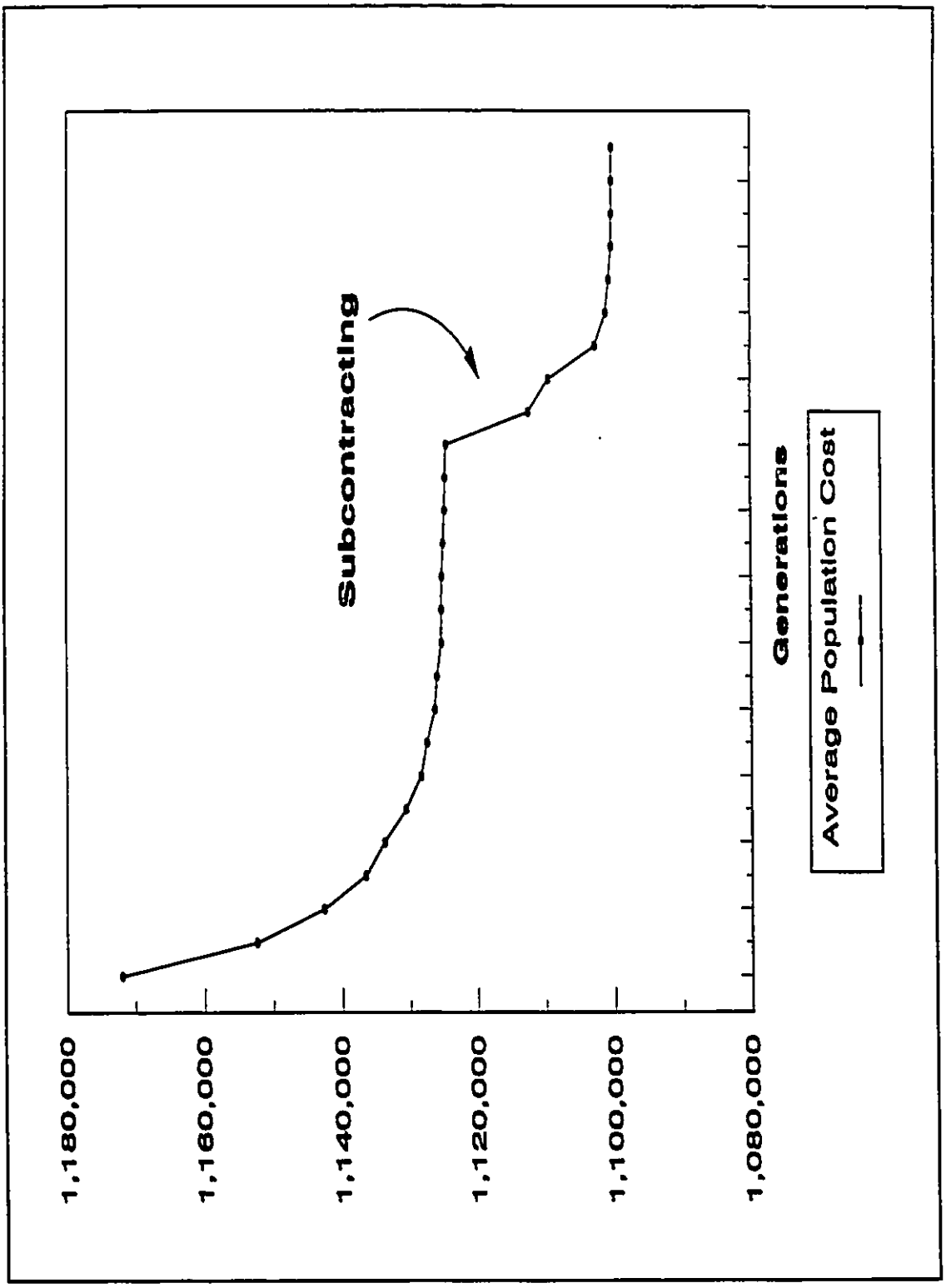

Figure 5.6. Average Population Cost Under GENMOD IV

Existence of a higher number of exceptional part types in periods 1 and 3 resulted in higher intercellular movement cost for GENMOD IV. Even though GENMOD IV obtained optimal results for five cost functions because of the higher intercellular movement cost, the overall total cost function value under GENMOD IV became sub-optimal.

The overall system utilization under both of the models is equal, since in each period the total number of machines from each type is the same. However, the utilization of each machine cell during periods shows variation under each model. The utilization of the machine cells under both of the models is given in Table 5.57.

The results in Table 5.57 indicate that for each period during the planning horizon, GENMOD IV achieved better machine utilizations than Model IV by assigning the machine capacities in a more balanced way. The computational time to reach the optimal solution under Model IV was 36:30 hours, whereas the computational time to reach the sub-optimal solution was 1:06 hours, *(96.9 % reduction in computational time).*

Even if the results under GENMOD IV are sub-optimal, the results encourage the use of GENMOD IV because of its capability to achieve balanced machine utilizations and to obtain an acceptable solution in a lesser amount of time as compared to Model IV.

Table 5.57. Machine Cell Utilization Under GENMOD IV and Model IV

		GENMOD IV		MODEL IV	
		Cell 1	Cell 2	Cell 1	Cell 2
Period I	Total Machining Capacity in the Cell	10,000	10,000	10,000	10,000
	Total Processing Time Within the Cell	8,469	5,841	8,933	5,377
	Total Idle Time	1,531	4,159	1,067	4,623
	Cell Utilization	0.85	0.584	0.89	0.54
	System Utilization	0.715		0.715	
Period II	Total Machining Capacity in the Cell	10,000	8,000	10,000	8,000
	Total Processing Time within the Cell	8,074	3,421	6,981	4,514
	Total Idle Time	1,926	4,579	3,019	3,486
	Cell Utilization	0.801	0.43	0.70	0.56
	System Utilization	0.639		0.639	
Period III	Total Machining Capacity in the Cell	8,000	8,000	10,000	6,000
	Total Processing Time within the Cell	5,399	4,058	5,471	3,986
	Total Idle Time	2,601	3,942	4,529	2,014
	Cell Utilization	0.675	0.51	0.55	0.66
	System Utilization	0.591		0.591	

5.8. Consistency of Genetic Algorithms

In this section, the generalized single period genetic algorithm, GENMOD II, is tested by using two different sets of data and the results are compared with the generalized single period mathematical model, Model II. The objective is to observe GENMOD II's accuracy in finding the optimal solution under different number of part machine combinations.

5.8.1. Example 1

Example 1 includes 12 different part types and 8 different machine types. During the calculations 2 machine cell formation is considered and for each intercellular movement, a cost of $2,000 is charged. Machine requirements, annual demand and unit subcontracting cost of the part types are given in Table 5.58. The additional cost parameters used in the calculations are also given in Table 5.59. To test GENMOD II, 60000 generations are created and after the 40000^{th} generation, subcontracting is applied. Each generation included a population of size 60. During the calculations, the probability of crossover and probability of mutation is taken as 0.89 and 0.089, respectively. The resulting part family and machine cell formations obtained from Model II are given in Tables 5.60 and 5.61, respectively, whereas the ones obtained from

GENMOD II are given in Tables 5.62 and 5.63, respectively. The comparison of the cost functions under both of the techniques is given in Table 5.64.

Table 5.58. Machine Requirements, Annual Demand and Subcontracting Cost of Part Types

Part Type	Machine Types								Annual Demand	Cost Value
	1	2	3	4	5	6	7	8		
1	0	2.3	0	0	2.6	0	0	2.4	17000	4.4
2	1.8	0	2.6	0	0	1.2	0	0	19000	3.9
3	1.9	0	0	0	0	2.5	2.8	0	18000	3.9
4	2.1	0	2.3	0	0	0	2.6	0	20000	4.1
5	0	2.6	1.5	0	0	0	0	2.3	16000	4.0
6	0	2.8	2.2	0	0	0	0	0	14000	0.4
7	2.3	0	2.8	0	0	2.6	0	0	20000	4.2
8	0	1.8	0	2.0	0	0	0	2.2	19000	4.0
9	0	2.0	0	2.4	0	0	2.6	0	21000	4.3
10	0	2.5	2.7	0	0	0	0	0	17000	3.6
11	2.3	0	0	0	2.1	0	2.4	0	19000	4.5
12	1.7	0	2.6	1.9	0	0	0	0	16000	3.5

Table 5.59. Machining, Capital and Idle Time Costs

Machine Type	Machining Cost (S/hr)	Capital Cost (S/year)	Idle Time Cost(S/hr)
1	6	18000	4
2	10	19000	2
3	8	22000	3
4	7	18000	5
5	9	21000	3
6	7	19000	3
7	6	20000	4
8	8	17000	5

Table 5.60. Part Family Formations Under Model II (Example 1)

Machining Cells	Part Families	
	Part Type	Total
Cell 1	1,5,7,8,9,10,11,12	8
Cell 2	2,3,4,7	4

Table 5.61. Machine Cell Composition Under Model II (Example 1)

Machining Cells	Machine Type								Total Number of Machines
	1	2	3	4	5	6	7	8	
Cell 1	1	2	1	1	1	0	1	1	8
Cell 2	1	0	2	0	0	1	1	0	5

Table 5.62. Part Family Formation Obtained Under GENMOD II (Example 1)

Machining Cells	Part Families	
	Part Type	Total
Cell 1	2,3,7,11	4
Cell 2	1,4,5,8,9,10,11,12	8

Table 5.63. Machine Cell Formation Obtained Under GENMOD II (Example1)

Machining Cells	Machine Type								Total Number of Machines
	1	2	3	4	5	6	7	8	
Cell 1	1	0	1	0	0	1	1	0	4
Cell 2	1	2	2	1	1	0	1	1	9

Table 5.64. Cost Functions for MODEL II and GENMOD II (Example 1)

Cost Functions	Model II	GENMOD II
Machine Investment Cost	255,000	255,000
Idle Time Cost	11,529	11,529
Machining Cost	167,590	167,590
Interceilular Movement Cost	2,000	2,000
Subcontracting Cost	5,600	5,600
Total Cost	**441,719**	**441,719**

The results indicate that both of the techniques created two part families with one exceptional part type, which is part type 7 under Model II and part type 11 under GENMOD II. Part type 6 is subcontracted under both of the techniques based on economic cost trade-offs. GENMOD II found the same optimal total cost values obtained from Model II. The graphical illustration of the average population cost value during generations, is given in Figure 5.7. The decreasing slope of the average cost of curve is an indication of GENMOD II's capability to generate part family and machine cell formations with lower cost values at each generation.

5.8.2. Example 2

Example 2 includes 15 different part types and 8 different machine types. During

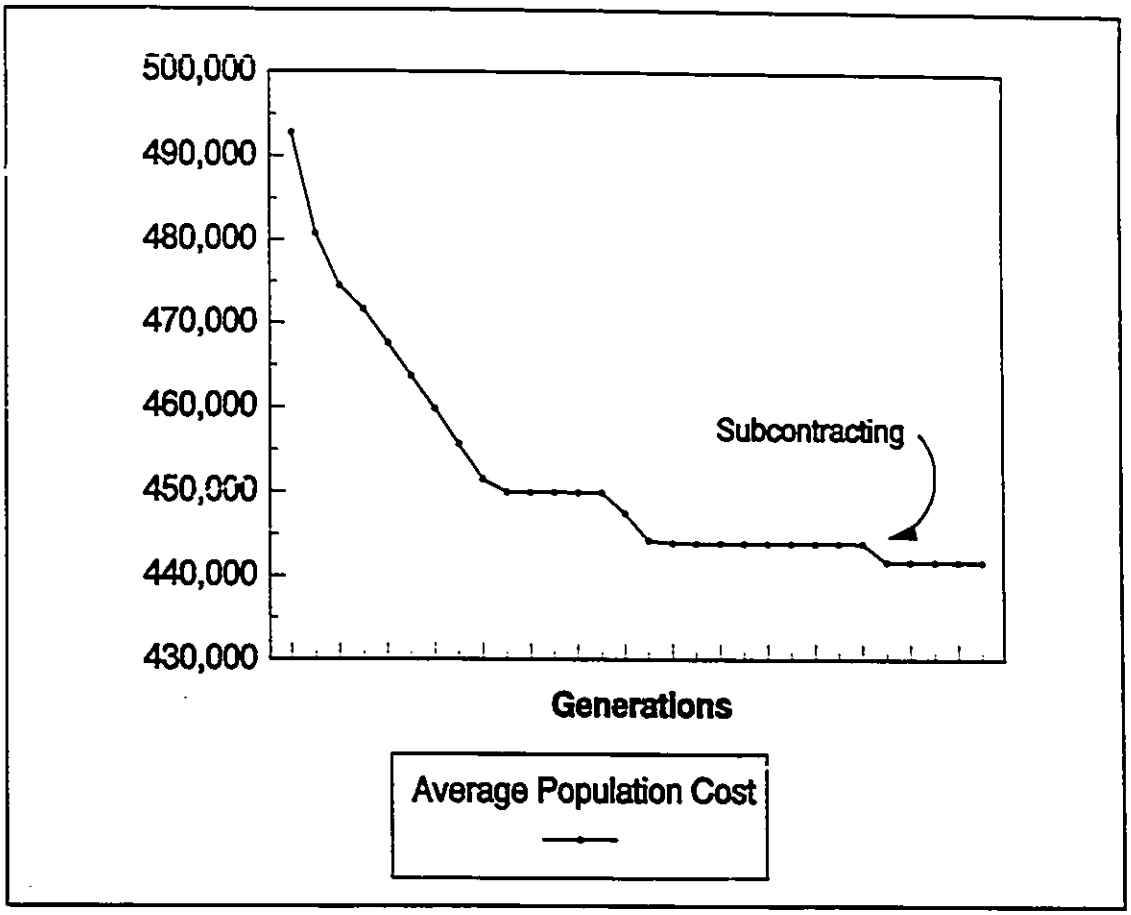

Figure 5.7. Average Population Cost Under GENMOD II (Example 1)

the calculations, 2 machine cell formation is considered and for each intercellular movement, a cost of $2,000 is charged. Machine requirements, annual demand and unit subcontracting cost of the part types are given in Table 5.65. The additional cost parameters used in the calculations are also given in Table 5.66. To test GENMOD II, 80000 generations are created and after the 60000th generation, subcontracting is applied. The population size, probability of crossover and mutation are taken the same as in Example 1. The resulting part family and machine cell formations obtained from

Model II are given in Tables 5.67 and 5.68, respectively, whereas the ones obtained from GENMOD II are given in Tables 5.69 and 5.70, respectively. Finally, the comparison of the cost functions under both of the techniques are given in Table 5.71.

Table 5.65. Machine Requirements, Annual Demand and Subcontracting Costs of Part Types

Part Type	Machine Types										Annual Demand	Subcontracting Cost($/unit)
	1	2	3	4	5	6	7	8	9	10		
1	1.8	0	0	0	2.0	0	2.4	0	2.2	0	17000	4.4
2	2.0	0	2.7	0	0	0	0	2.1	0	1.6	23000	4.3
3	1.6	0	1.5	0	0	0	0	1.8	0	0	15000	4
4	0	0	2.5	0	0	0	2.6	0	2.7	0	21000	4.4
5	0	0	2.3	0	0	2.8	0	0	0	0	20000	4.3
6	2.4	0	0	0	2.2	0	0	0	0	0	18000	4.2
7	0	2.0	0	0	0	2.2	0	0	0	2.1	21000	4.6
8	0	1.5	0	0	0	2.9	0	1.4	0	1.7	22000	4.1
9	0	0	0	2.0	0	2.4	0	0	0	0	16000	0.45
10	0	2.5	0	0	0	3.2	0	2.8	0	3.1	18000	3.8
11	0	1.8	0	0	0	3.0	0	0	2.3	0	16000	3.9
12	0	2.7	0	2.1	0	0	0	2.2	0	2.4	17000	4.3
13	0	1.9	0	0	2.1	0	0	1.8	2.3	0	22000	3.9
14	2.3	0	0	2.4	0	0	2.7	0	2.1	0	21000	4.1
15	2.1	0	2.6	0	0	0	2.8	0	2.4	0	19000	4.2

Table 5.66. Machining, Capital and Idle Time Costs

Machine Type	Machining Cost($/hr)	Capital Cost($/year)	Idle Time Cost($/hr)
1	8	18000	3
2	9	20000	4
3	7	17000	5
4	10	22000	2
5	9	21000	4
6	9	18000	2

7	10	23000	3
8	8	21000	2
9	7	19000	4
10	9	20000	5

Table 5.67. Part Family Formation Under Model II (Example 2)

Machining Cells	Part Families	
	Part Type	Total
Cell 1	3,5,10,11,12,**14,15**	7
Cell 2	1,2,4,6,7,8,13,**14,15**	9

Table 5.68. Machine Cell Formation Under Model II (Example 2)

Machining Cells	Machine Type										Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	
Cell 1	1	1	1	1	0	2	0	1	1	1	9
Cell 2	1	1	1	0	1	1	2	1	2	1	11

Table 5.69. Part Family Formation Under GENMOD II (Example 2)

Machining Cells	Part Families	
	Part Type	Total
Cell 1	2,3,5,8,10,11,12,**13**,14,15	10
Cell 2	1,2,4,6,7,8,**10,13**	8

Table 5.70. Machine Cell Formation Under GENMOD II (Example 2)

Machining Cells	Machine Type										Total Number of Machines
	1	2	3	4	5	6	7	8	9	10	
Cell 1	1	1	1	1	0	2	1	2	2	1	12
Cell 2	1	1	1	0	1	1	1	0	1	1	8

Table 5.71. Cost Functions Under Model II and GENMOD II (Example 2)

Cost Functions	Model II	GENMOD II
Machine Investment Cost	392,000	392,000
Idle Time Cost	15,332	15,332
Machining Cost	295,772	295,772
Intercellular Movement Cost	4,000	8,000
Subcontracting Cost	7,200	7,200
Total Cost	**714,304**	**718,304**

The results indicate that both of the models created two part families with exceptional parts. Model II created two exceptional part types, which are: 14 and 15, whereas GENMOD II created four exceptional part types, which are: 2, 8, 10 and 13. Part type 9 is subcontracted under both of the techniques based on the economic cost trade-offs. For this particular example, GENMOD II found the sub-optimal total cost value with an error of 0.56%. The graphical illustration of the average population cost value during generations, is given in Figure 5.8. The decreasing slope of the average cost curve is an indication of GENMOD II's capability to generate part family and

machine cell formations with lower cost values at each generation.

The results obtained from the previous examples and the two current ones (Examples 1 & 2) indicate that genetic algorithms can find the solutions for the previously developed mathematical models with minimal error and they can be used as a reliable tool for finding optimal or sub-optimal solutions for the optimization problems.

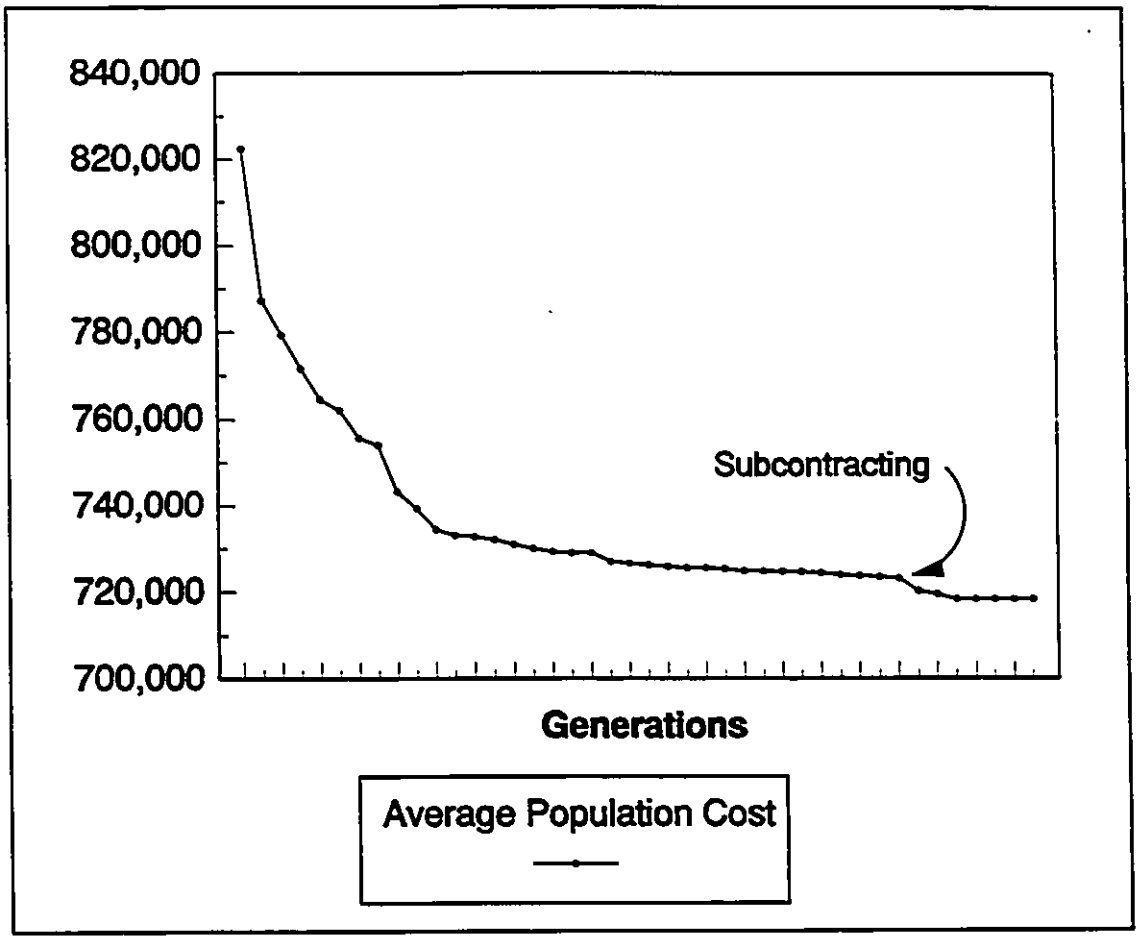

Figure 5.8. Average Population Cost Under GENMOD II (Example 2)

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

6.1. Conclusions

In this research, mathematical models are developed to simultaneously develop part family and machine cell formations under single and multi period planning horizons. The objective of the models is to minimize the system cost while attempting to obtain independent machine cells and balanced machining capacity distribution.

The single period mathematical models consider the economic cost trade-offs among intercellular movement, machine duplication and subcontracting, whereas the multi period mathematical models consider the economic cost trade-offs among intercellular movement, machine duplication, subcontracting and machine relocation, for each period in the planning horizon.

A heuristic technique, based on the mathematical models, is developed by using Genetic Algorithms to reduce the amount of computational time for large size problems. The heuristic technique considered the cost functions and operating constraints specified in the mathematical models.

The mathematical models are tested by using different sizes of numerical examples. Single period mathematical models created intercellular movements instead of duplicating the bottleneck machines and subcontracted some of the parts based on the cost trade-offs. Multi period mathematical models achieved a balanced machining capacity distribution by changing the configuration of the machine cells so that the variation in the number of machines from each type is minimized among periods. It is concluded that all the mathematical models achieved independent machine cells based on economic cost trade-offs.

A different type of multi period planning strategy is proposed, which pushes the same machine cell configuration formed in the first period throughout the planning horizon without recognizing the possibility of fluctuations in part demand and changes in machine cell configurations in future periods. The proposed strategy is tested by using the same examples used to test the multi period mathematical models. It was observed that the strategy resulted in flexible machine cells with low machine cell utilization. Also, under the proposed strategy the capital investment cost and idle time costs were higher. It is concluded that consideration of the fluctuations in part demand in future periods, during the planning stage resulted in manufacturing systems with higher machine cell utilizations and lower system costs.

The heuristic techniques, GENMOD I and GENMOD II, developed under single period planning horizon, are tested by using a large size example and the results

are compared with single period mathematical models. It was observed that the heuristic method had the capability of finding a sub-optimal solution with a 0.49 % error by reducing the computational time by at least 90 %, as compared to the single period mathematical models. Finally, GENMOD II is tested by using two additional data sets. For the first data set, it found the optimal solution, whereas for the second data set it found the sub-optimal solution with an error of 0.56%. These results supported the efficiency and accuracy of the heuristic technique. Then, the heuristic techniques, GENMOD III and GENMOD IV, developed under multi period planning horizon are tested by using the same example used to test the multi period mathematical models. The results indicated that the heuristic method had the capability of finding a sub-optimal solution with less than 2 % error by achieving more balanced machine cell utilizations in less computational time, as compared to the multi period mathematical models.

6.2. Future Research

In the mathematical models several cost functions are considered such as capital investment cost, intercellular movement cost, idle time cost, etc. Capital tool investment cost and intra cellular movement cost can be included.

It was assumed that each part had only one process plan. This assumption can

be relaxed by considering multiple process plans for each part type. Additionally, subcontracting was allowed, but the whole demand must be subcontracted. This assumption can also be relaxed by considering the option of subcontracting a portion of the given demand for a particular part type.

The demand of the part types and processing times on the machines were taken as deterministic values. A simulation model can be developed to observe the behaviour of the mathematical models under stochastic environments.

REFERENCES

Adil, G. K., Rajamani, D. and Strong, D., "A Mathematical Model for Cell Formation Considering Investment and Operational Costs", *European Journal of Operational Research*, 69, pp 330-341, 1993.

Askin, R. G. and Chiu, K. S., "A Graph Partitioning Procedure for Machine Assignment and Cell Formation in Group Technology", *International Journal of Production Research*, Vol. 28, No. 8, pp 1555-1572, 1990.

Askin, R. G., Cresswell, S. H., Goldberg, J. B. and Vakharia, A. J., "A Hamiltonian Path Approach to Reordering the Part-Machine Matrix for Cellular Manufacturing", *International Journal of Production Research*, Vol. 29, No. 6, pp 1081-1100, 1991.

Atmani, A., Laskhari, R. S. and Caron, R. J., "A Mathematical Programming Approach to Joint Cell Formation and Operation Allocation in Cellular Manufacturing", to appear in *International Journal of Production Research*, 1994.

Atmani, A., Laskhari, R. S. and Caron, J., "Joint Cell Formation and Operation Allocation in Manufacturing Systems with Machine Cost Considerations", to appear in *ORSA The Institute of Management Sciences Joint Annual Meeting*, October 1994.

Austih, S., "An Introduction to Genetic Algorithms", *AI Expert*, pp 48-55, 1990.

Ben-Arieh, D. and Chang, P. T., "An Extension to the p-Median Group Technology Algorithm", *Computers and Operations Research*, Vol. 21, No. 2, pp 119-125, 1994.

Black, J. T., "The Design of the Factory With A Future", *McGraw-Hill, Inc*, Reading, 1991.

Boe, W. J. and Cheng, C. H., "A Close Neighbour Algorithm for Designing Cellular Manufacturing Systems", *International Journal of Production Research*, Vol. 29, No. 10, pp 2097-2116, 1991.

Burbidge, J. L., "Production Flow Analysis", *The Production Engineer*, pp 139-152, April-May, 1971.

Burbidge, J. L., "Production Flow Analysis for Planning Group Technology", *Clarendon Press*, Reading, Oxford, 1989.

Burbidge, J. L., "Change to Group Technology: Process Organization is Obsolete", *International Journal of Production Research*, Vol. 30, No. 5, pp 1209-1219, 1992.

Chan, H. M. and Milner, D. A., "Direct Clustering Algorithm for Group Formation in Cellular Manufacture", *Journal of Manufacturing Systems*, Vol. 1, No. 1, pp 65-75, 1982.

Chandrasekharan, M. P. and Rajagopalan, R., "MODROC: An Extension of Rank Order Clustering for Group Technology", *International Journal of Production Research*, Vol. 24, No. 5, pp 1221-1233, 1986.

Chen, H. G. and Guerrero, H. H., "A General Search Algorithm for Cell Formation in Group Technology", *International Journal of Production Research*, Vol. 32, No. 11, pp 2711-2724, 1994.

Choobineh, F., "A Framework for the Design of Cellular Manufacturing Systems", *International Journal of Production Research*, Vol. 26, No. 7, pp 1161-1172, 1988.

Chu, C. and Tsai, M., "A Comparison of Three Array-Based Clustering Techniques for Manufacturing Cell Formation", *International Journal of Production Research*, Vol. 28, No. 8, pp 1417-1433, 1990.

Chow, W. S. and Hawaleshka, O., "Minimizing Intercellular Part Movements in Manufacturing cell Formation", *International Journal of Production Research*, Vol. 31, No. 9, pp 2161-2170, 1993.

Co, H. C. and Araar, A., "Configuring Cellular Manufacturing Systems", *International Journal of Production Research*, Vol. 26, No. 9, pp 1511-1522, 1988.

Dahel, N. E. and Smith, S. B., "Designing Flexibility into Cellular Manufacturing Systems", *International Journal of Production Research*, Vol. 31, No. 4, pp 933-945, 1993.

Damodran, V., Lashkari, R. S. and Singh, N., "A Production Planning Model for Cellular Manufacturing Systems with Refixturing Considerations", *International Journal of Production Research*, Vol. 30, No. 7, pp 1603-1615, 1992.

De Witte, J., "The Use of Similarity Coefficients in Production Flow Analysis", *International Journal of Production Research*, Vol. 18, No. 4, pp 503-514, 1980.

Ferreira Ribeiro, J. F. and Pradin, B., "A Methodology for Cellular Manufacturing Design", *International Journal of Production Research*, Vol. 31, No. 1, pp 235-250, 1993.

Ganesh, M. V. and Srinivasan, G., "A Heuristic Algorithm for the Cell Formation Problem", *Computers and Industrial Engineering*, Vol. 26, No. 1, pp 193-201, 1994.

Glover, F., "Tabu Search - Part I", *ORSA Journal on Computing*, Vol. 1, No. 3, pp 190-206, 1989.

Glover, F., "Tabu Search - Part II", *ORSA Journal on Computing*, Vol 2, No. 1, pp 4-32, 1990.

Goldberg, D. E., "Genetic Algorithms in Search Optimization and Machine Learning", *Addison-Wesley Publishing Company, Inc.*, Reading, Massachusetts, 1987.

Gunasingh, K. R. and Lashkari, R. S., "The Cell Formation Problem In Cellular Manufacturing Systems - A Sequential Modelling Approach", *Computers and Industrial Engineering*, Vol. 16, No. 4, pp 469-476, 1989.

Gunasingh, K. R. and Lashkari, R. S., "Machine grouping problem in Cellular Manufacturing Systems - an Integer Programming Approach", *International Journal of Production Research*, Vol. 27, No. 9, pp 1465-1473, 1989.

Gunasingh, K. R. and Lashkari, R. S., "Simultaneous Grouping of Parts and Machines in Cellular Manufacturing Systems - An Integer Programming Approach", *Computers and Industrial Engineering*, Vol. 20, No. 1, pp 111-117, 1991.

Gupta, T., "Design of Manufacturing Cells for Flexible Environment Considering Alternative Routing", *International Journal of Production Research*, Vol. 31, No. 6, pp 1259-1273, 1993.

Gupta, T. and Seifoddini, H., "Production Data Based Similarity Coefficient for Machine-Component Grouping Decisions in the Design of a Cellular Manufacturing System", *International Journal of Production Research*, Vol. 28, No. 7, pp 1247-1269, 1990.

Heragu, S. S. and Gupta, Y. P., "A Heuristic for Designing Cellular Manufacturing Facilities", *International Journal of Production Research*, Vol. 32, No. 1, pp 125-140, 1994.

Jain, A. K., Kasilingam, R. G. and Bhole, S. D., "Joint Consideration of Cell Formation and Tool Provisioning Problems in Flexible Manufacturing Systems", *Computers and Industrial Engineering*, Vol. 20, No. 2, pp 271-277, 1991.

Kern, G. M. and Wei, J. C., " The Cost of Eliminating Exceptional Elements in Group Technology Cell Formation", *International Journal of Production Research*, Vol. 29, No. 8, pp 1535-1547, 1991.

Khator, S. K. and Irani, S. A., "Cell Formation in Group Technology: A New Approach", *Computers and Industrial Engineering*, Vol. 12, No. 2, pp 131-142, 1987.

King, J. R., "Machine Component Grouping in Production Flow Analysis: An Approach Using a Rank Order Clustering Algorithm", *International Journal of Production Research*, Vol. 18, No. 2, pp 213-232, 1980.

King, J. R. and Nakornchai V., "Machine-Component Group Formation in Group Technology: Review and Extension", *International Journal of Production Research*, Vol. 20, No. 2, pp 117-133, 1982.

Kumar, R. V. and Vannelli A., "Strategic Subcontracting for Efficient Disaggrated Manufacturing", *International Journal of Production Research*, Vol. 25, No. 12, pp 1715-1728, 1987.

Kusiak, A., "The Generalized Group Technology Concept", *International Journal of Production Research*, Vol. 25, No. 4, pp 561-569, 1987.

Kusiak, A. and Chow, W. S., "Efficient Solving of the Group Technology Problem", *Journal of Manufacturing Systems*, Vol. 6, No. 2, pp 117-124, 1987.

Lam, J. and Delosme, J.- M., "Performance of a New Annealing Schedule", *25th ACM/IEEE Desing Automation Conference*, pp 306-311, 1988.

Lee, C. S. and Hwang, H., "A Hierarchial Divisive Clustering Method for Machine-Component Grouping Problems", *Engineering Optimization*, Vol. 17, pp 65-78, 1991.

Lee, H. and Garcia-Diaz, A., "A Network Flow Approach to Solve Clustering Problems in Group Technology", *International Journal of Production Research*, Vol. 31, No. 3, pp 603-612, 1993.

Liang, M., "Integrating Machining Speed, Part Selection and Machine Loading Decisions in Flexible Manufacturing Systems", *Computers and Industrial Engineering*, Vol. 26, No. 3, pp 599-608, 1994.

Liang, M. and Taboun, S., "Part Selection and Part Assignment in Flexible Manufacturing Systems with Cellular Layout", *Computers and Industrial Engineering*, Vol. 23, Nos. 1-4, pp 63-67, 1992.

Liao, W. T., "Design of Line-Type Cellular Manufacturing Systems for Minimum Operating and Material-Handling Costs", *International Journal of Production Research*, Vol. 32, No. 2, pp 387-397, 1994.

Logendran, R., "A Workload Based Model for Minimizing Total Intercell and Intracell Moves in Cellular Manufacturing", *International Journal of Production Research*, Vol. 28, No. 5, pp 913-925, 1990.

Logendran, R. and West, T. M., "A Machine-Part Based Grouping Algorithm in Cellular Manufacturing", *Computers and Industrial Engineering*, Vol. 19, Nos. 1-4, pp 57-61, 1990.

Logendran, R., "Effect of the Identification of Key Machines in the Cell Formation Problem of Cellular Manufacturing Systems", *Computers and Industrial Engineering*, Vol. 20, No. 4, pp 439-449, 1991.

Logendran, R., "A Binary Integer Programming Approach for Simultaneous Machine-Part Grouping in Cellular Manufacturing Systems", *Computers and Industrial Engineering*, Vol. 24, No. 3, pp 329-336, 1993.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E., "Equation of State Calculations by Fast Computing Machines", *Journal of Chemical Physics*, Vol. 21, pp 1087-1092, 1953.

McAuley, J., "Machine Grouping for Efficient Production", *The Production Engineer*, pp 53-57, February, 1972.

Okogbaa, O. G., Chen, M., Changchit, C. and Shell, R. L., "Manufacturing System Cell Formation and Evaluation Using a New Inter-Cell Flow Reduction Heuristic", *International Journal of Production Research*, Vol. 30, No. 5, pp 1101-1118, 1992.

Oliveria, J. F. C. and Ferreira, J. A. S., "Algorithms for Nesting Problems", *Lecture Notes in Economics and Mathematical Systems*, Vol. 396, pp 256-273, 1993.

Paulli, J., "A Computational Comparison of Simulated Annealing and Tabu Search Applied to the Quadratic Assignment Problem", *Lecture Notes in Economics and Mathematical Systems*, Vol. 396, pp 86-102, 1993.

Rajagopalan, R. and Batra, J. L., "Design of Cellular Production Systems: A Graph Teoric Approach", *International Journal of Production Research*, Vol. 13, No. 6, pp 567-579, 1975.

Rajamani, D., Singh, N. and Aneja, Y. P., "Integrated Design of Cellular Manufacturing Systems in the Presence of Alternative Process Plans", *International Journal of Production Research*, Vol. 28, No. 8, pp 1541-1554, 1990.

Rajamani, D., Singh, N. and Aneja, Y. P., "A Model for Cell Formation in Manufacturing Systems with Sequence Dependence", *International Journal of Production Research*, Vol. 30, No. 6, pp 1227-1235, 1992.

Rodrigues, M. R. D. and Anjo, A. J. B., "On Simulating Thermodynamics", *Lecture Notes in Economics and Mathematical Systems*, Vol. 396, pp 46-60, 1993.

Sankaran, S., "Multiple Objective Decision Making Approach to Cell Formation: A Goal Programming Model", *Mathematical Computation and Modelling*, Vol. 13, No. 9, pp 71-82, 1990.

Sankaran, S. and Kasilingam, R. G., "An Integrated Approach to Cell Formation and Part Routing in Group Technology Manufacturing Systems", *Engineering Optimization*, Vol. 16, pp 235-245, 1990.

Seifoddini, H. and Hsu, C., "Comparative Study of Similarity Coefficients and Clustering Algorithms in Cellular Manufacturing", *Journal of Manufacturing Systems*, Vol. 13, No. 2, pp 119-127, 1994.

Shafer, S. M., Kern, G. M. and Wei, J. C., "A Mathematical Programming Approach for Dealing with Exceptional Elements in Cellular Manufacturing", *International Journal of Production Research*, Vol. 30, No. 5, pp 1029-1036, 1992.

Shafer, S. M. and Rogers, D. F., "Similarity and Distance Measures for Cellular Manufacturing. Part I. A Survey", *International Journal of Production Research*, Vol. 31, No. 5, pp 1133-1142, 1993.

Shafer, S. M. and Rogers, D. F., "Similarity and Distance Measures for Cellular Manufacturing. Part II. An Extension and Comparison", *International Journal of Production Research*, Vol. 31, No. 6, pp 1315-1326, 1993.

Shtub, A., "Modelling Group Technology Cell Formation as a Generalized Assignment Problem", *International Journal of Production Research*, Vol. 27, No. 5, pp 775-782, 1989.

Steudel, J. H. and Ballakur, A., "A Dynamic Programming Based Heuristic for Machine Grouping in Manufacturing Cell Formation", *Computers and Industrial Engineering*, Vol. 12, No. 3, pp 215-222, 1987.

Suer, G. A. and Ortega, M., "A Machine Level Based-Similarity Coefficient for Forming Manufacturing Cells", *Computers and Industrial Engineering*, Vol. 27, Nos. 1-4, pp 67-70, 1994.

Sundaram, R. M., "Cellular Manufacturing Systems Design With Alternative Routing Considerations", *Journal of Mechanical Working Technology*, Vol. 20, pp 425-432, 1989.

Taboun, S. M., Sankaran, S. and Bhole, S., "Comparison and Evaluation of Similarity Measures in Group Technology", *Computers and Industrial Engineering*, Vol. 20, No. 3, pp 343-353, 1991.

Taboun, S. M. and Sharma, A., "A Weighted Index for the Design of Cellular Manufacturing Systems", *Computers and Industrial Engineering*, Vol. 21, Nos. 1-4, pp 273-277, 1991.

Valle, A. G. D., Balarezo, S. and Tejero, J., "A Heuristic Workload-Based Model to Form Cell by Minimizing Intercellular Movements", *International Journal of Production Research*, Vol. 32, No. 10, pp 2275-2285, 1994.

Vohra, T., Chen, D., Chang, J. C. and Chen, H., " A Network Approach to Cell Formation in Cellular Manufacturing", *International Journal of Production Research*, Vol. 28, No. 11, pp 2075-2084, 1990.

Waghodekar, P. H. and Sahu, S., "Machine-Component Cell Formation in Group Technology: MACE", *International Journal of Production Research*, Vol. 22, No. 6, pp 937-948, 1984.

Wei, J. C. and Kern, G. M., "Commonality Analysis: A Linear Cell Clustering Algorithm for Group Technology", *International Journal of Production Research*, Vol. 27, No. 12, pp 2053-2062, 1989.

Wemmerlow, U. and Hyer, N. L., "Cellular Manufacturing in the U.S. Industry: A Survey of Users", *International Journal of Production Research*, Vol. 27, No. 9, pp 1511-1530, 1989.

Wu, N. and Salvendy, G., "A Modified Network Approach for the Design of Cellular Manufacturing Systems", *International Journal of Productions Research*, Vol. 31, No. 6, pp 1409-1421, 1993.

APPENDIX I

Source Code of the Genetic Algorithms

```
Program GENMOD I;

Uses crt;

Const
 MaxOperation = 114;          {Number of operations * CN}
 Part = 20;                   {Total number of parts}
 Operation = 57;              {Number of operations}
 MCType = 12;                 {Actual # of Machines Types}
 Pop = 60;                    {Population Size}
 CN = 2;                      {Number of Machine Cells}
 MaxGen = 100000;             {Maximum number of generations}
 PMutation = 0.089;           {Probability of Mutation}
 PCrossOver = 0.89;           {Probability of CrossOver}
 pop2 = pop+2;
 Cap = 2000;

Type
 Costarray = array[1..MCType] of real;
 Testarray = array[1..Part] of integer;
 Timearray = array[1..Part,1..MCType] of integer;
 Chromosome = array[1..MaxOperation] of integer;
 MachineConfiguration = array[1..MCType,1..CN] of integer;
 Idlearray = array[1..MCType,1..CN] of integer;
 Individual = Record
         chrom : Chromosome;      {The string structure}
         fitness : real;          {Objective function Value}
         chance : boolean;        {true if less than the average}
         pselect : real;          {prob of selection to next gen.}
        end;
 Population = array[1..Pop2] of Individual;

Var
 ns,ns1,ns2,ns3,ns4,ns5 : real;
 IdCost, Capital: CostArray; {Arrays that contain the cost values}
 Test, OpNo : Testarray;
 Time : Timearray; {Array that stores the production requirements of parts}
 Capacity, MacNo : MachineConfiguration; {Arrays that store the number of
 machines in each machine cell}
 Idle : Idlearray; {Array that stores the idle time of each machine in each cell}
 MacFile : Text;
 Resultfile, Dummyfile, Infile, Outfile : Text; {Input & Output Files}
 Name : String[15];
 m, i, j, k, num : integer;
 Number : Real; {obj value}
 Dummypop, Oldpop : Population;
 Avg, Avg1, Avg2, SumFitness, SumFitness1 : Real;
 Gen : longint;  {Index for number of generations}
 nmutation, ncrossover : real; {total # of mutation and crossover occured}
 Min, Max : real;
 Minimum1, Maximum1 : integer;
 ok:boolean; {Boolean Variable to show if the machine cells are within limits}

{********** PROCEDURE DECLERATION **********************}

 {**********PROCEDURE INITMACHINE**********************}
{This procedure initializes contents of all the necessary arrays to zero}
Procedure InitMachine;
Var
 p,p1 : integer;

begin
 For p:=1 to MCType do
  For p1:=1 to CN do
   begin
    MacNo[p,p1]:=0;
    Idle[p,p1]:=0;
    Capacity[p,p1]:=0;
   end;
end;

        {**********PROCEDURE RUNLINDO************}
Procedure RunLindo (var RLINDO: individual);
```

```
var
 r1,r2,r3,r4,r5:integer;
 move : real;
begin
 number:=0;
 ns:=0;
 ns1:=0;
 ns2:=0;
 ns3:=0;

 {Calculation of the Capital Investment Cost}
 For r1:=1 to MCType do
  For r2:=1 to CN do
   ns:=ns+(Capital[r1]*MacNo[r1,r2]);

 {Calculation of the Idle Time Cost}
 For r1:=1 to MCType do
  For r2:=1 to CN do
   ns1:=ns1+(IdCost[r1]*Idle[r1,r2]);

 {Calculation of the Intercellular Movement Cost}
 Move:=0.0;
 For r1:=1 to CN do
  begin
   r4:=0;
   For r2:=1 to Part do
    begin
     r3:=0;
     Repeat
      r3:=r3+1;
      r4:=r4+1;
      r5:=r4+(operation*(r1-1));
      If (RLindo.chrom[r5]-RLindo.chrom[r5+1])=1 then
       Move:=Move+1.0;
     Until (r3)=(OpNo[r2]-1);
     r4:=r4+1;
    end;
  end;
 ns3:=2000*Move;
 number:=ns+ns1+ns3;
end;

      {**********PROCEDURE CALCULATESTAT**********}
Procedure CalculateStat;
Var
 st1, selected :integer;
begin
 sumfitness:=0.0;   avg:=0.0;
 Sumfitness:=oldpop[1].fitness;
 Min:=oldpop[1].fitness;
 Max:=oldpop[1].fitness;
 Maximum1:=1;
 Minimum1:=1;
 For st1:=2 to pop2 do with oldpop[st1] do
  begin
   Sumfitness:=Sumfitness+fitness;
   If fitness>max then
    begin
     max:=fitness;
     maximum1:=st1;
    end;
   If fitness<min then
    begin
     min:=fitness;
     minimum1:=st1;
    end;
  end;
 avg:=(sumfitness/pop2);
 sumfitness1:=0.0;
 avg1:=0.0;
 selected:=0;

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
```

```
   If (fitness <= avg) then
    begin
     chance:=true;
     sumfitness1:=sumfitness1+fitness;
     selected := selected + 1;
    end;
   If (fitness > avg) then
    chance:=false;
  end;
 avg1:=(sumfitness1/selected);

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
   If chance then
    pselect:=((2*avg1-fitness)/sumfitness1);
   If not(chance) then
    pselect:=0;
  end;

end; {end of calculatestat}

      {**********FUNCTION FLIP**********}
Function flip(probability:real):integer;
Var
 t:real;

begin
 If (probability=1) then
  flip:=1;
 If (probability<>1) then
  begin
   t:=random;
   If (t<=probability) then
    flip:=1
   else
    flip:=0;
  end;
end;

      {**********FUNCTION MUTATION**********}
Function Mutation(Mut:integer):integer;
Var
 DoMutation, Dum1 : Integer;
begin
 DoMutation:=flip(pmutation);
 If (DoMutation=1) then
  begin
   nmutation:=nmutation+1;
   If (Mut=1) then
    Dum1:=0;
   If (Mut=0) then
    Dum1:=1;
  end;
 If (DoMutation=0) then
  Dum1:=Mut;
 Mutation:=Dum1;
end;

      {**********PROCEDURE REPRODUCE**********}
Procedure Reproduce;
Var
 r1,r2,full : integer;
 pselect : real;

begin
 full:=0;
 r1:=0;
 Repeat
  r1:=r1+1;
  If (oldpop[r1].chance) then
   r2:=flip(oldpop[r1].pselect);
  If (r2=1) and (oldpop[r1].chance) then
   begin
    full:=full+1;
```

```
    dummypop[full]:=oldpop[r1];
   end;
  If (r1=pop2) then
   r1:=0;
 Until (full=pop2);
 oldpop:=dummypop;
end;

{***********PROCEDURE PARAMETERCALCULATION***********}
Procedure ParameterCalculation (var paracal : individual);
Var
 k,k1,k2,k3,k4,k5,cap1 : integer;

begin
 k2:=0;
 For k:=1 to Part do
  begin
   For k1:=1 to MCType do
    begin
     If (Time[k,k1]) > 0 then
      begin
       k2:=k2+1;
       For k3:=1 to CN do
        begin
         k4:=k2+(operation*(k3-1));
         If (paracal.chrom[k4]=1) then
          k5:=k3;
        end;
       Capacity[k1,k5]:=Capacity[k1,k5]+Time[k,k1];
      end; {if time[k,k1]>0}
    end; {for k1:=1 to mctype}
  end;  {for k:=1 to part}
 {Calculation of the number of machines from each type in each machine cell}
 For k:=1 to MCType do
  begin
   For k1:=1 to CN do
    begin
     If (Capacity[k,k1]=0) then
      MacNo[k,k1]:=0;
     If (Capacity[k,k1]>0) then
      begin
       Cap1:=Capacity[k,k1] div cap;
       If (Capacity[k,k1] mod cap = 0) then
        MacNo[k,k1]:=cap1;
       If (Capacity[k,k1] mod cap > 0) then
        MacNo[k,k1]:=cap1+1;
      end; { if cap>0}
    end; {k1=1 to CN}
  end; {k=1 to MCtype}
 {Calculation of the idle time of the machines in the machine cells}
 For k:=1 to MCType do
  begin
   For k1:=1 to CN do
    begin
     Idle[k,k1]:=(Cap*MacNo[k,k1])-Capacity[k,k1];
    end;
  end;
end; {Parameter Calculation}

        {***********PROCEDURE FEASIBILITY***********}
Procedure Feasibility (var feasible : individual);
var
 h,h1,h2,h3,h4,excess,totalop : integer;
begin
 For h:=1 to Operation do
  begin
   totalop:=0;
   For h1:=1 to CN do
    begin
     h2:=h+(operation*(h1-1));
     totalop:=totalop+feasible.chrom[h2];
    end;
   If (totalop=0) then
    begin
```

```
      h3:=random(CN)+1;
      h4:=h+(operation*(h3-1));
      feasible.chrom[h4]:=1;
     end;
    If (totalop > 1) then
     begin
      excess:=totalop-1;
      h1:=0;
      Repeat
       h2:=random(CN)+1;
       h3:=h+(operation*(h2-1));
       If (feasible.chrom[h3]=1) then
        begin
         h1:=h1+1;
         feasible.chrom[h3]:=0;
        end;
      Until (excess=h1);
     end;
   end; {h=1 to operation}
end;

        {************PROCEDURE CROSSOVER**************}
Procedure CrossOver(l:integer);
Var
 Mate1, Mate2, DoCrossOver, Crosssite, cr1:integer;
begin
 DoCrossOver:=flip(pcrossover);
 Mate1:=random(pop)+1;  {Randomly select two mates among the members of
 the population}
 Mate2:=random(pop)+1;

 If (DoCrossOver=1) then
  begin
   ncrossover:=ncrossover+1;
   crosssite:=random(MaxOperation);
   If (crosssite=0) then
    crosssite:=crosssite+1;
  end;
 If (DoCrossOver=0) then
  crosssite:=MaxOperation;

 For cr1:=1 to crosssite do
  begin
   oldpop[l].chrom[cr1]:=mutation(oldpop[mate1].chrom[cr1]);
   oldpop[l+1].chrom[cr1]:=mutation(oldpop[mate2].chrom[cr1]);
  end;
 If (crosssite <> MaxOperation) then
  begin
   For cr1:=(crosssite+1) to MaxOperation do
    begin
     oldpop[l].chrom[cr1]:=mutation(oldpop[mate2].chrom[cr1]);
     oldpop[l+1].chrom[cr1]:=mutation(oldpop[mate1].chrom[cr1]);
    end;
  end;
 Feasibility(oldpop[l]),
 Feasibility(oldpop[l+1]);
end;

        {************PROCEDURE DATA INPUT**************}
{This procedure stores production requirements of the parts and all other cost
values into related arrays}

Procedure DataInput;
begin
 Write('Time Data File==> ');
 Readln(Name);
 Assign(MacFile,Name);
 Reset(MacFile);
 For i:=1 to Part do
  begin
   OpNo[i]:=0;
   For k:=1 to MCType do
    begin
```

```
      Read(MacFile,num);
      Time[i,k]:=num;
      If (num<>0) then
       begin
        OpNo[i]:=OpNo[i]+1;
        Test[i]:=k;
       end;
     end;
    Readln(MacFile);
  end;
Close(MacFile);

Write('Enter the Idle time file==> ');
Readln(Name);
Assign(Macfile,Name);
Reset(MacFile);
For i:=1 to MCType do
 begin
  Readln(Macfile,num);
  Idcost[i]:=num;
 end;
Close(MacFile);

Write('Enter the Capital Cost file==> ');
Readln(Name);
Assign(Macfile,Name);
Reset(MacFile);
For i:=1 to MCType do
 begin
  Readln(Macfile,num);
  Capital[i]:=num;
 end;
Close(MacFile);

end;

      {**********PROCEDURE MACHINECHECK**********}

Procedure MachineCheck;

Var
 pm,pm1,pm2:integer;

begin
 For pm:=1 to CN do
  begin
   pm2:=0;
   For pm1:=1 to MCType do
    begin
     pm2:=pm2+MacNo[pm1,pm];
    end;
    If (pm2 <9) then
     begin
      ok:=false;
     end;
  end;
end;

          {**********PROCEDURE INITDATA**********}
Procedure InitData;
Var
 j, j1,j2,j3,j4,j5,j6 : integer;

begin
 For j:=1 to Pop2 do with oldpop[j] do
  begin
   Repeat
    ok:=true;
    For j1:=1 to MaxOperation do
     chrom[j1]:=0;
    j4:=0;

    For j1:=1 to Part do
     begin
      For j2:=1 to MCType do
```

```
        begin
         If (Time[j1,j2]>0) then
          begin
           j3:=Random(CN)+1;
           j4:=j4+1;
           For j5:=1 to CN do
            begin
             j6:=j4+(operation*(j5-1));
             If (j5=j3) then
              chrom[j6]:=1;
             If (j5<>j3) then
              chrom[j6]:=0;
            end;
          end;
        end;
      end;
     Initmachine;
     ParameterCalculation(oldpop[j]);
     MachineCheck;
    Until ok;

     Runlindo(oldpop[j]);
     Oldpop[j].fitness:=number;
     Writeln('pop ',j,' ',number:10:3);
  end; {for j:=1 to pop}
end;


              {*******PROCEDURE ARRAYSORT********}
Procedure ArraySort;
var
 NoSwap, Pass, First : integer;
 TempRec : Individual;
begin
 Pass:=1;
 Repeat
  Noswap:=0;
  For First:=1 to (Pop2-Pass) do
   begin
    If (oldpop[first].fitness > oldpop[first+1].fitness) then
     begin
      Temprec:=oldpop[first];
      oldpop[first]:=oldpop[first+1];
      oldpop[first+1]:=Temprec;
      NoSwap:=NoSwap+1;
     end;
   end;
  Pass:=Pass+1;
 Until (Noswap=0);

end;

     {**********PROCEDURE CALCULATE AVERAGE**********}
{This procedure calculates the average fitness value of the population}
Procedure CalculateAverage;
var
ca: integer;
tot: real;
begin
 avg2:=0;
 tot:=0;
 For ca:=1 to Pop do
  tot:=tot+oldpop[ca].fitness;
 avg2:=(tot/pop);
end;

          {************ PROCEDURE REPORT*************}
Procedure Report;
begin
 for i:=1 to Pop2 do
  begin
   Writeln(Resultfile,'Final pop: ',i,' ',oldpop[i].fitness:12:3);
   Initmachine;
   ParameterCalculation(oldpop[i]);
```

```
  Runlindo(oldpop[i]);
  Writeln(resultfile,number:10:1,'/',ns:10:1,'/',ns1:10:1,'/',ns2:10:1,'/',ns3:10:1);
  For m:=1 to CN do
  For j:=1 to MCType do
   begin
    Write(resultfile,MacNo[j,m],',');
    if j=mctype then
     Writeln(resultfile);
   end;
 Writeln(resultfile);
 For m:=1 to CN do
  for j:=1 to MCType do
   Write(resultfile,idle[j,m],',');
 Writeln(resultfile);
 For k:=1 to MaxOperation do
  begin
   Write(resultfile,oldpop[i].chrom[k],',');
    if k=operation then Writeln(resultfile);
  end;
 Writeln(Resultfile);
end;

 Writeln(resultfile,'mutation: ',nmutation:10:1);
 Writeln(resultfile,'crossover:',ncrossover:10:1);
 Close(Resultfile);
end;

              {**********MAIN PROGRAM**********}

begin
 ClrScr;
 Randomize;
 Assign(Resultfile,'Res1.dat');
 Rewrite(Resultfile);
 NMutation:=0;
 NCrossOver:=0;
 gen:=0;
 DataInput;
 Initdata;
 arraysort;
 CalculateStat;
 reproduce;
 arraysort;
 clrscr;
Repeat
 Gen:=Gen+1;
 Repeat
  gotoxy(15,4);
  ok:=true;
  Crossover(pop+1);
  Initmachine;
  ParameterCalculation(oldpop[pop+1]);
  MachineCheck;
  Initmachine;
  ParameterCalculation(oldpop[pop+2]);
  MachineCheck;
 Until ok;

 For k:=1 to 2 do
  begin
   Initmachine;
   ParameterCalculation(oldpop[pop+k]);
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
   oldpop[pop+k].chance:=true;
   oldpop[pop+k].pselect:=0.5;
  end;
 Write('GENERATION ',gen,' Is Complete');
 arraysort;
 gotoxy(15,5);
 Write('The best Found:  ',oldpop[1].fitness:12:1);
 if (gen mod 1000) = 0 then
  begin
   calculateaverage;
```

```
    Writeln(resultfile,'gen : ',gen,'  average: ',avg2:10:3);
    Writeln(resultfile,'First: ',oldpop[1].fitness:10:3,'
    //Last',oldpop[pop].fitness:10:3);
   end;
Until (gen=maxgen);

Report;
end. {Main Program}
```

Program GENMOD II;

```
uses crt;

Const
 MaxOperation = 114;              {Number of operations * CN}
 Part = 20;                       {Total number of parts}
 Operation = 57;                  {Number of operations}
 MCType = 12;                     {Actual # of Machines Types}
 Pop = 60;                        {Population Size}
 CN = 2;                          {Number of Machine Cells}
 MaxGen = 60000;                  {Maximum number of generations w/o
                                   subcontracting}
 FinalGen = 100000;               {Maximum number of generations for the
                                   whole}
 PMutation = 0.089;               {Probability of Mutation}
 PCrossOver = 0.89;               {Probability of CrossOver}
 pop2 = pop+2;
 Cap = 2000;
 maxop = maxoperation+part;
Type
 Subarray = array[1..Part] of real;
 Costarray = array[1..MCType] of real;
 Testarray = array[1..Part] of integer;
 Timearray = array[1..Part,1..MCType] of integer;
 Chromosome = array[1..MaxOp] of integer;
 MachineConfiguration = array[1..MCType,1..CN] of integer;
 Idlearray = array[1..MCType,1..CN] of integer;
 Individual = Record
         chrom : Chromosome;
         fitness : real;          {Objective function Value}
         chance : boolean;        {true if less than the average}
         pselect : real;          {prob of selection to next gen.}
        end;
 Population = array[1..Pop2] of Individual;

Var
 sub: subarray;
 num1,de:real;
 ns,ns1,ns2,ns3,ns4,ns5 : real;
 IdCost, Capital, Production : CostArray; {Arrays that contain the cost values}
 Test, OpNo : Testarray;
 Time : Timearray; {Array that store the production requirements of parts}
 Capacity, MacNo : MachineConfiguration; {Arrays that store the number of
 machines in each cell}
 Idle : Idlearray; {Array that stores the idle time of each machine in each cell}
 MacFile : Text;
 Resultfile, Dummyfile, Infile, Outfile : Text; {Input & Output Files}
 Name : String[15];
 m, i, j, k, num : integer;
 Number : Real; {obj value}
 Dummypop, Oldpop : Population;
 Avg, Avg1, Avg2, SumFitness, SumFitness1 : Real;
 Gen : longint;  {Index for number of generations}
 nmutation, ncrossover : real; {Total # of mutation and crossover occured}
 Min, Max : real;
 best,Minimum1, Maximum1 : integer;
 ok:boolean;

   {*******PROCEDURE DECLERATION**********************}

    {***************PROCEDURE  INITMACHINE*************}
Procedure InitMachine;
Var
 p,p1 : integer;

begin
 For p:=1 to MCType do
  For p1:=1 to CN do
   begin
    MacNo[p,p1]:=0;
    Idle[p,p1]:=0;
    Capacity[p,p1]:=0;
```

```
    end;
end;

        {**************PROCEDURE RUNLINDO****************}

Procedure RunLindo (var RLINDO: individual);

var
 r1,r2,r3,r4,r5:integer;
 move : real;
begin
 number:=0;
 ns:=0;
 ns1:=0;
 ns2:=0;
 ns3:=0;
 ns4:=0;


 {Calculation of the Capital Investment Cost}
 For r1:=1 to MCType do
  For r2:=1 to CN do
   ns:=ns+(Capital[r1]*MacNo[r1,r2]);

 {Calculation of the Idle Time Cost}
 For  r1:=1 to MCType do
  For r2:=1 to CN do
   ns1:=ns1+(IdCost[r1]*Idle[r1,r2]);

 {Calculation of the Machining Cost}
 r3:=0;
 For r1:=1 to Part do
  begin
   For r2:=1 to MCType do
    begin
     If (Time[r1,r2] > 0) then
      begin
       r3:=r3+1;
       For r4:=1 to CN do
        begin
         r5:=r3+(operation*(r4-1));
         ns2:=ns2+(RLINDO.chrom[r5]*(Time[r1,r2]*Production[r2]));
        end;
      end;
    end;
  end;

{Calculation of the Intercellular Movement Cost}
Move:=0.0;
For r1:=1 to CN do
 begin
  r4:=0;
  For r2:=1 to Part do
   begin
    r3:=0;
    Repeat
     r3:=r3+1;
     r4:=r4+1;
     r5:=r4+(operation*(r1-1));
     If (RLindo.chrom[r5]-RLindo.chrom[r5+1])=1 then
      Move:=Move+1.0;
    Until (r3)=(OpNo[r2]-1);
    r4:=r4+1;
   end;
 end;
ns3:=2000*Move;

{Calculation of the Subcontracting Cost}
For r1:=1 to Part do
 begin
  ns4:=ns4+(Rlindo.chrom[maxoperation+r1]*Sub[r1]);
 end;
number:=ns+ns1+ns2+ns3+ns4;
end;
```

```
    {***************PROCEDURE CALCULATESTAT***********}
Procedure CalculateStat;
Var
 st1, selected :integer;
begin
 sumfitness:=0.0;   avg:=0.0;
 Sumfitness:=oldpop[1].fitness;
 Min:=oldpop[1].fitness;
 Max:=oldpop[1].fitness;
 Maximum1:=1;
 Minimum1:=1;
 For st1:=2 to pop2 do with oldpop[st1] do
  begin
   Sumfitness:=Sumfitness+fitness;
   If fitness>max then
    begin
     max:=fitness;
     maximum1:=st1;
    end;
   If fitness<min then
    begin
     min:=fitness;
     minimum1:=st1;
    end;
  end;
 avg:=(sumfitness/pop2);
 sumfitness1:=0.0;
 avg1:=0.0;
 selected:=0;

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
   If (fitness <= avg) then
    begin
     chance:=true;
     sumfitness1:=sumfitness1+fitness;
     selected:=selected + 1;
    end;
   If (fitness > avg) then
    chance:=false;
  end;
 avg1:=(sumfitness1/selected);

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
   If chance then
    pselect:=((2*avg1-fitness)/sumfitness1);
   If not(chance) then
    pselect:=0;
  end;
end; {end of calculatestat}

            {***********FUNCTION FLIP***********}
Function flip(probability:real):integer;
Var
 t:real;

begin
 If (probability=1) then
  flip:=1;
 If (probability<>1) then
  begin
   t:=random;
   If (t<=probability) then
    flip:=1
   else
    flip:=0;
  end;
end;

          {**********FUNCTION MUTATION***********}
Function Mutation(Mut:integer):integer;
Var
 DoMutation, Dum1 : Integer;
```

```
begin
 DoMutation:=flip(pmutation);
 If (DoMutation=1) then
  begin
   nmutation:=nmutation+1;
   If (Mut=1) then
    Dum1:=0;
   If (Mut=0) then
    Dum1:=1;
  end;
 If (DoMutation=0) then
  Dum1:=Mut;
 Mutation:=Dum1;
end;

        {************PROCEDURE REPRODUCE*************}
Procedure Reproduce;
Var
 r1,r2,full : integer;
 pselect : real;

begin
 full:=0;
 r1:=0;
 Repeat
  r1:=r1+1;
  If (oldpop[r1].chance) then
   r2:=flip(oldpop[r1].pselect);
  If (r2=1) and (oldpop[r1].chance) then
   begin
    full:=full+1;
    dummypop[full]:=oldpop[r1];
   end;
  If (r1=pop2) then
   r1:=0;
 Until (full=pop2);
 oldpop:=dummypop;
 end;

{**************PROCEDURE PARAMETERCALCULATION************}
 Procedure ParameterCalculation (var paracal : individual);
 Var
  k,k1,k2,k3,k4,k5,cap1 : integer;

 begin
  k2:=0;
  For k:=1 to Part do
   begin
    For k1:=1 to MCType do
     begin
      If (Time[k,k1]) > 0 then
       begin
        k2:=k2+1;
        For k3:=1 to CN do
         begin
          k4:=k2+(operation*(k3-1));
          Capacity[k1,k3]:=Capacity[k1,k3]+(Time[k,k1]*paracal.chrom[k4]);
         end;
       end; {if time[k,k1]>0}
     end; {for k1:=1 to mctype}
   end;  {for k:=1 to part}

  {Calculation of the number of machines from each type in each machine cell}
  For k:=1 to MCType do
   begin
    For k1:=1 to CN do
     begin
      If (Capacity[k,k1]=0) then
       MacNo[k,k1]:=0;
      If (Capacity[k,k1]>0) then
       begin
        Cap1:=Capacity[k,k1] div cap;
        If (Capacity[k,k1] mod cap = 0) then
         MacNo[k,k1]:=cap1;
```

```
        If (Capacity[k,k1] mod cap > 0) then
          MacNo[k,k1]:=cap1+1;
       end; { if cap>0}
     end; {k1=1 to CN}
  end; {k=1 to MCtype}

 {Calculation of the idle time of the machines in the machine cells}
 For k:=1 to MCType do
  begin
   For k1:=1 to CN do
    begin
     Idle[k,k1]:=(Cap*MacNo[k,k1])-Capacity[k,k1];
    end;
  end;
end; {Parameter Calculation}

      {**************PROCEDURE FEASIBILITY***************}
Procedure Feasibility (var feasible : individual);
var
 h,h1,h2,h3,h4,h5,excess,totalop : integer;
begin
 For h:=1 to Operation do
  begin
   totalop:=0;
   For h1:=1 to CN do
    begin
     h2:=h+(operation*(h1-1));
     totalop:=totalop+feasible.chrom[h2];
    end;
   If (totalop=0) then
    begin
     h3:=random(CN)+1;
     h4:=h+(operation*(h3-1));
     feasible.chrom[h4]:=1;
    end;
   If (totalop > 1) then
    begin
     excess:=totalop-1;
     h1:=0;
     Repeat
      h2:=random(CN)+1;
      h3:=h+(operation*(h2-1));
      If (feasible.chrom[h3]=1) then
       begin
        h1:=h1+1;
        feasible.chrom[h3]:=0;
       end;
     Until (excess=h1);
    end;
  end; {h=1 to operation}

 For h:=1 to Part do
  begin
   If (feasible.chrom[maxoperation+h]=1) then
    begin
     h2:=0;
     For h1:=1 to (h-1) do
      h2:=h2+OpNo[h1];
     For h3:=1 to CN do
      begin
       For h4:=1 to OpNo[h] do
        begin
         h5:=h2+h4+(operation*(h3-1));
         feasible.chrom[h5]:=0;
        end;
      end;
    end;
  end;   {for h:=1 to Part}

end; {feasibility}


          {***********PROCEDURE CROSSOVER***********}
Procedure CrossOver(l:integer);
```

```
Var
 Mate1, Mate2, DoCrossOver, Crosssite, cr1,rf,rf1:integer;
begin

 DoCrossOver:=flip(pcrossover);
 {Randomly select two mates among the members of the population}
 Mate1:=random(pop)+1;
 Mate2:=random(pop)+1;

 If (DoCrossOver=1) then
  begin
   ncrossover:=ncrossover+1;
   crosssite:=random(MaxOperation);
   If (crosssite=0) then
    crosssite:=crosssite+1;
  end;
 If (DoCrossOver=0) then
  crosssite:=MaxOperation;

 For cr1:=1 to crosssite do
  begin
   oldpop[l].chrom[cr1]:=mutation(oldpop[mate1].chrom[cr1]);
   oldpop[l+1].chrom[cr1]:=mutation(oldpop[mate2].chrom[cr1]);
  end;
 If (crosssite <> MaxOperation) then
  begin
   For cr1:=(crosssite+1) to MaxOperation do
    begin
     oldpop[l].chrom[cr1]:=mutation(oldpop[mate2].chrom[cr1]);
     oldpop[l+1].chrom[cr1]:=mutation(oldpop[mate1].chrom[cr1]);
    end;
  end;

 For cr1:=1 to Part do
  begin
        oldpop[l].chrom[maxoperation+cr1]:=oldpop[mate1].
        chrom[maxoperation+cr1];
        oldpop[l+1].chrom[maxoperation+cr1]:=oldpop[mate2].
        chrom[maxoperation+cr1];
  end;
 Feasibility(oldpop[l]);
 Feasibility(oldpop[l+1]);
end;

          {********PROCEDURE DATA INPUT**********}
{This procedure stores production requirements of the parts and all other cost
values into related arrays}
Procedure DataInput;
begin
 Write('Time Data File==> ');
 Readln(Name);
 Assign(MacFile,Name);
 Reset(MacFile);
 For i:=1 to Part do
  begin
   OpNo[i]:=0;
   For k:=1 to MCType do
    begin
     Read(MacFile,num);
     Time[i,k]:=num;
     If (num<>0) then
      begin
       OpNo[i]:=OpNo[i]+1;
       Test[i]:=k;
      end;
    end;
   Readln(MacFile);
  end;
 Close(MacFile);

 Write('Enter the Idle time file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
```

```
For i:=1 to MCType do
 begin
  Readln(Macfile,num1);
  Idcost[i]:=num1;
 end;
Close(MacFile);

Write('Enter the Capital Cost file==> ');
Readln(Name);
Assign(Macfile,Name);
Reset(MacFile);
For i:=1 to MCType do
 begin
  Readln(Macfile,num1);
  Capital[i]:=num1;
 end;
Close(MacFile);

Write('Enter the Production Cost file==> ');
Readln(Name);
Assign(Macfile,Name);
Reset(MacFile);
For i:=1 to MCType do
 begin
  Readln(Macfile,num1);
  Production[i]:=num1;
 end;
Close(MacFile);

Write('Enter the Subcontracting Cost File==> ');
Readln(Name);
Assign(MacFile,Name);
Reset(MacFile);
For i:=1 to Part do
 begin
  Readln(MacFile,num1);
  Sub[i]:=num1;
  end;
 Close(MacFile);
end;

      {**********PROCEDURE MACHINECHECK**********}
Procedure MachineCheck;
Var
 pm,pm1,pm2:integer;

begin
 For pm:=1 to CN do
  begin
   pm2:=0;
   For pm1:=1 to MCType do
     pm2:=pm2+MacNo[pm1,pm];
    If (pm2 <9) then
      ok:=false;
  end;
end;

        {**********PROCEDURE  FINDTHEBEST**********}
Procedure FindTheBest (var subrecord:individual);
var
 srecord : individual;
 s1,s2,s3,s4,s5,s6,pf,pf1 : integer;
 num2,num3:real;
begin

 num2:=subrecord.fitness;
 num3:=subrecord.fitness;

 For s1:=1 to Part do
  begin
   srecord:=subrecord;
   ok:=true;
   s3:=0;
   srecord.chrom[maxoperation+s1]:=1;
```

```
    For s2:=1 to (s1-1) do
     s3:=s3+OpNo[s2];
    For s4:=1 to CN do
     begin
      For s5:=1 to OpNo[s1] do
       begin
        s6:=s3+s5+(operation*(s4-1));
        srecord.chrom[s6]:=0;
       end;
     end;
    Initmachine;
    ParameterCalculation(srecord);
    machinecheck;
    If ok then
     begin
      Runlindo(srecord);
      If (number < num2) and (number < num3) then
       begin
        best:=s1;
        num3:=number;
       end;
     end;
    srecord.chrom[maxoperation+s1]:=0;
   end;
end;

      {*******PROCEDURE SUBCONTRACT THE PART********}
Procedure SubcontractThePart (var stprecord:individual);
var
 st1,st2,st3,st4 : integer;

begin
 st1:=0;
 For st2:=1 to (best-1) do
  st1:=st1+OpNo[st2];
 For st2:=1 to CN do
  begin
   For st3:=1 to OpNo[best] do
    begin
     st4:=st1+st3+(operation*(st2-1));
     stprecord.chrom[st4]:=0;
    end;
  end;
 stprecord.chrom[maxoperation+best]:=1;
end;

            {*********PROCEDURE INITDATA*********}
Procedure InitData;
Var
 j, j1,j2,j3,j4,j5,j6 : integer;

begin
 For j:=1 to Pop2 do with oldpop[j] do
  begin
   Repeat
    ok:=true;
    For j1:=1 to (MaxOp) do
     chrom[j1]:=0;
    j4:=0;

    For j1:=1 to Part do
     begin
      For j2:=1 to MCType do
       begin
        If (Time[j1,j2]>0) then
         begin
          j3:=Random(CN)+1;
          j4:=j4+1;
          For j5:=1 to CN do
           begin
            j6:=j4+(operation*(j5-1));
            If (j5=j3) then
             chrom[j6]:=1;
            If (j5<>j3) then
```

```
            chrom[j6]:=0;
           end;
         end;
       end;
     end;
    Initmachine;
    ParameterCalculation(oldpop[j]);
    MachineCheck;
   Until ok;

    Runlindo(oldpop[j]);
    Oldpop[j].fitness:=number;
    Writeln('pop ',j,' ',number:10:3);
  end; {for j:=1 to pop}
end;

      {************PROCEDURE ARRAYSORT***************}
Procedure ArraySort;
var
 NoSwap, Pass, First : integer;
 TempRec : Individual;
begin
 Pass:=1;
 Repeat
  Noswap:=0;
  For First:=1 to (Pop2-Pass) do
   begin
    If (oldpop[first].fitness > oldpop[first+1].fitness) then
     begin
      Temprec:=oldpop[first];
      oldpop[first]:=oldpop[first+1];
      oldpop[first+1]:=Temprec;
      NoSwap:=NoSwap+1;
     end;
   end;
  Pass:=Pass+1;
 Until (Noswap=0);
end;

   {**********PROCEDURE CALCULATE AVERAGE*************}
{This procedure calculates the average fitness value of the population}
Procedure CalculateAverage;
var
 ca: integer;
 tot: real;
begin
 avg2:=0;
 tot:=0;
 For ca:=1 to Pop do
  tot:=tot+oldpop[ca].fitness;
 avg2:=(tot/pop);
end;

          {********** PROCEDURE REPORT *************}
Procedure Report;
begin
 for i:=1 to Pop2 do
  begin
   Writeln(Resultfile,'Final pop: ',i,'  ',oldpop[i].fitness:12:3);
   Initmachine;
   ParameterCalculation(oldpop[i]);
   Runlindo(oldpop[i]);
   Writeln(resultfile,number:10:1,'/',ns:10:1,'/',ns1:10:1,'/',
   ns2:10:1,'/',ns3:10:1,'/',ns4:10:1);
  For m:=1 to CN do
   For j:=1 to MCType do
    begin
     Write(resultfile,MacNo[j,m],',');
     if j=mctype then
      Writeln(resultfile);
    end;
  Writeln(resultfile);
  For m:=1 to CN do
   for j:=1 to MCType do
```

```
    Write(resultfile,idle[j,m],',');
    Writeln(resultfile);

  For k:=1 to MaxOperation+part do
   begin
    Write(resultfile,oldpop[i].chrom[k],',');
    if (k=operation) or (k=maxoperation) then
    Writeln(resultfile);
   end;
  Writeln(Resultfile);
end;

Writeln(resultfile,'mutation: ',nmutation:10:1);
Writeln(resultfile,'crossover:',ncrossover:10:1);
Close(Resultfile);
end;

            {**************MAIN PROGRAM***************}

begin
 ClrScr;
 Randomize;
 Assign(Resultfile,'Res1.dat');
 Rewrite(Resultfile);
 NMutation:=0;
 NCrossOver:=0;
 gen:=0;
 DataInput;
 Initdata;
 calculatestat;
 reproduce;
 arraysort;
 clrscr;
 Repeat
 Gen:=Gen+1;
  Repeat
   ok:=true;
    Crossover(pop+1);
    Initmachine;
    ParameterCalculation(oldpop[pop+1]);
    MachineCheck;
  Initmachine;
  ParameterCalculation(oldpop[pop+2]);
  MachineCheck;
 Until ok;

 For k:=1 to 2 do
  begin
   Initmachine;
   ParameterCalculation(oldpop[pop+k]);
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
   oldpop[pop+k].chance:=true;
   oldpop[pop+k].pselect:=0.5;
  end;
 gotoxy(15,4);
 Writeln('GENERATION ',gen,' Is Complete');
 arraysort;
 if (gen mod 1000) = 0 then
  begin
   calculateaverage;
   Writeln(resultfile,'gen: ',gen,' average: ',avg2:10:3);
   Writeln(resultfile,'first\oldpop[1].fitness:10:3,'//last',oldpop[pop].fitness:10:3);
  end;
Until (gen=maxgen);

{Subcontracting Begins}
For i:=1 to Pop do
 begin
  Repeat
   best:=part+1;
   FindTheBest(oldpop[i]);
   If ((Best<>(Part+1)) and (oldpop[i].chrom[maxoperation+Best]<>1)) then
    begin
```

```
      SubcontractThePart(oldpop[i]);
      Initmachine;
      ParameterCalculation(oldpop[i]);
      RunLindo(oldpop[i]);
      oldpop[i].fitness:=number;
    end;
  Until (Best=part+1);
end;
arraysort;

Repeat
 Gen:=Gen+1;
 Repeat
  ok:=true;
  Crossover(pop+1);
  Initmachine;
  ParameterCalculation(oldpop[pop+1]);
  MachineCheck;
  Initmachine;
  ParameterCalculation(oldpop[pop+2]);
  MachineCheck;
 Until ok;

 For k:=1 to 2 do
  begin
   Initmachine;
   ParameterCalculation(oldpop[pop+k]);
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
   oldpop[pop+k].chance:=true;
   oldpop[pop+k].pselect:=0.5;
  end;

 gotoxy(15,4);
 Writeln('GENERATION ',gen,' Is Complete');
 arraysort;
 if (gen mod 1000) = 0 then
   begin
    calculateaverage;
    Writeln(resultfile,'gen: ',gen,' average: ',avg2:10:3);
      Writeln(resultfile,'First:     ',oldpop[1].fitness:10:3,'
//Last',oldpop[pop].fitness:10:3);
   end;

Until (gen=finalgen);
Report;
end. {Main Program}
```

Programs GENMOD III;

```
uses crt;

Const
 MaxOperation = 48;              {Number of operations * CN}
 Part = 10;                      {Total number of parts}
 Operation = 24;                 {Number of operations}
 MCType = 7;                     {Actual # of Machines Types}
 Pop = 60;                       {Population Size}
 CN = 2;                         {Number of Machine Cells}
 MaxGen = 90000;                 {Maximum number of generations}
 PMutation = 0.04955;            {Probability of Mutation}
 PCrossOver = 0.9989;            {Probability of CrossOver}
 pop2 = pop+2;
 Period = 3;
 Cap = 2000;

Type
 Costarray = array[1..MCType] of real;
 Testarray = array[1..Part,1..Period] of integer;
 Timearray = array[1..Part,1..MCType,1..Period] of integer;
 Chromosome = array[1..Period,1..MaxOperation] of integer;
 MachineConfiguration = array[1..MCType,1..CN,1..Period] of integer;
 Idlearray = array[1..MCType,1..CN,1..Period] of integer;
 Individual = Record
         chrom : Chromosome;     {The string structure}
         fitness : real;         {Objective function Value}
         chance : boolean;       {true if less than the average}
         pselect : real;         {prob of selection to next gen.}
        end;
 Population = array[1..Pop2] of Individual;
Var
real;
 ns,ns1,ns2,ns3,ns4,ns5 : real;
 IdCost, Capital : CostArray; {Arrays that contain the cost values}
 Test, OpNo : Testarray;
 Time : Timearray; {Array that stores the production requirements of parts}
 Capacity, MacNo : MachineConfiguration; {Arrays that store the number of
 machines in each machine cell for each period}
 Idle : Idlearray; {Array that stores the idle time of each machine in each cell for
 each period}
 MacFile : Text;
 Resultfile, Dummyfile, Infile, Outfile : Text; {Input & Output Files}
 Name : String[15];
 m, t, i, j, k, num : integer;
 Number : Real; {obj value}
 Dummypop, Oldpop : Population;
 Avg, Avg1, Avg2, SumFitness, SumFitness1 : Real;
 Gen : longint;  {Index for number of generations}
 nmutation, ncrossover : real; {Total # of mutation and crossover occured}
 Min, Max : real;
 Minimum1, Maximum1 : integer;
 ok:boolean;

        {**********PROCEDURE DECLERATION**********}

    {**************PROCEDURE INITMACHINE***************}
Procedure InitMachine(var p2:integer);
{This procedure initializes the contents of all the necessary arrays to zero}
Var
 p,p1 : integer;

begin
 For p:=1 to MCType do
  For p1:=1 to CN do
   begin
    MacNo[p,p1,p2]:=0;
    Idle[p,p1,p2]:=0;
    Capacity[p,p1,p2]:=0;
   end;
```

```
end;

        {************PROCEDURE RUNLINDO***************}
Procedure RunLindo (var RLINDO: individual);
var
 r1,r2,r3,r4,r5,r6:integer;
 move : real;
begin
 number:=0;
 ns:=0;
 ns1:=0;
 ns2:=0;
 ns3:=0;
 ns4:=0;
 ns5:=0;

 {Calculation of the Capital Investment Cost}
 For r6:=1 to Period do
  For r1:=1 to MCType do
   For r2:=1 to CN do
    ns:=ns+(Capital[r1]*MacNo[r1,r2,r6]);

 {Calculation of the Idle Time Cost}
 For r6:=1 to Period do
  For  r1:=1 to MCType do
   For r2:=1 to CN do
    ns1:=ns1+(IdCost[r1]*Idle[r1,r2,r6]);

 {Calculation of the Intercellular Movement Cost}
 Move:=0.0;
 For r6:=1 to Period do
  begin
   For r1:=1 to CN do
    begin
     r4:=0;
     For r2:=1 to Part do
      begin
       r3:=0;
       Repeat
        r3:=r3+1;
        r4:=r4+1;
        r5:=r4+(operation*(r1-1));
         If (RLindo.chrom[r6,r5]-RLindo.chrom[r6,r5+1])=1 then
          Move:=Move+1.0;
       Until (r3)=(OpNo[r2,r6]-1);
       r4:=r4+1;
      end;
    end;
  end;
 ns3:=2000*Move;

 {Calculation of the Relocation Cost}
 For r6:=1 to (Period-1) do
  For r1:=1 to CN do
   For r2:=1 to MCType do
    begin
     ns4:=ns4+(5000*abs(MacNo[r2,r1,r6+1]-MacNo[r2,r1,r6]));
    end;
 number:=ns+ns1+ns3+ns4;

end;

        {**********PROCEDURE CALCULATESTAT***********}
Procedure CalculateStat;
Var
 st1, selected :integer;
begin
 sumfitness:=0.0;   avg:=0.0;
 Sumfitness:=oldpop[1].fitness;
 Min:=oldpop[1].fitness;
 Max:=oldpop[1].fitness;
 Maximum1:=1;
 Minimum1:=1;
 For st1:=2 to pop2 do with oldpop[st1] do
```

```
begin
 Sumfitness:=Sumfitness+fitness;
 If fitness>max then
  begin
   max:=fitness;
   maximum1:=st1;
  end;
 If fitness<min then
  begin
   min:=fitness;
   minimum1:=st1;
  end;
end;
avg:=(sumfitness/pop2);
sumfitness1:=0.0;
avg1:=0.0;
selected:=0;

For st1:=1 to pop2 do with oldpop[st1] do
 begin
  If (fitness <= avg) then
   begin
    chance:=true;
    sumfitness1:=sumfitness1+fitness;
    selected := selected + 1;
   end;
  If (fitness > avg) then
   chance:=false;
 end;
avg1:=(sumfitness1/selected);

For st1:=1 to pop2 do with oldpop[st1] do
 begin
  If chance then
   pselect:=((2*avg1-fitness)/sumfitness1);
  If not(chance) then
   pselect:=0;
  end;
end; {end of calculatestat}

                {**********FUNCTION FLIP**********}
Function flip(probability:real):integer;
Var
 t1 : real;

begin
 If (probability=1) then
  flip:=1;
 If (probability<>1) then
  begin
   t1:=random;
   If (t1<=probability) then
    flip:=1
   else
    flip:=0;
  end;
end;

        {**************FUNCTION MUTATION***************}
Function Mutation(Mut:integer):integer;
Var
 DoMutation, Dum1 : Integer;
begin
 DoMutation:=flip(pmutation);
 If (DoMutation=1) then
  begin
   nmutation:=nmutation+1;
   If (Mut=1) then
    Dum1:=0;
   If (Mut=0) then
    Dum1:=1;
  end;
 If (DoMutation=0) then
  Dum1:=Mut;
```

```
 Mutation:=Dum1;
end;

{************PROCEDURE REPRODUCE*****************}
Procedure Reproduce;
Var
 r1,r2,full : integer;
pselect : real;

begin
 full:=0;
 r1:=0;
 Repeat
  r1:=r1+1;
  If (oldpop[r1].chance) then
   r2:=flip(oldpop[r1].pselect);
  If (r2=1) and (oldpop[r1].chance) then
   begin
    full:=full+1;
    dummypop[full]:=oldpop[r1];
   end;
  If (r1=pop2) then
   r1:=0;
 Until (full=pop2);
 oldpop:=dummypop;
end;

{************PROCEDURE  PARAMETERCALCULATION************}
Procedure ParameterCalculation (var paracal : individual;
                    var perindex : integer);
Var
 k,k1,k2,k3,k4,k5,cap1 : integer;

begin
 k2:=0;
 For k:=1 to Part do
  begin
  For k1:=1 to MCType do
   begin
    If (Time[k,k1,perindex]) > 0 then
     begin
      k2:=k2+1;
      For k3:=1 to CN do
       begin
        k4:=k2+(operation*(k3-1));
        If (paracal.chrom[perindex,k4]=1) then
         k5:=k3;
       end;
       Capacity[k1,k5,perindex]:=Capacity[k1,k5,perindex]+
       Time[k,k1,perindex];
     end; {if time[k,k1]>0}
   end;  {for k1:=1 to mctype}
 end;  {for k:=1 to part}
{Calculation of the number of machines for a given period}
For k:=1 to MCType do
 begin
  For k1:=1 to CN do
   begin
    If (Capacity[k,k1,perindex]=0) then
     MacNo[k,k1,perindex]:=0;
    If (Capacity[k,k1,perindex]>0) then
     begin
      Cap1:=Capacity[k,k1,perindex] div cap;
      If (Capacity[k,k1,perindex] mod cap = 0) then
       MacNo[k,k1,perindex]:=cap1;
      If (Capacity[k,k1,perindex] mod cap > 0) then
       MacNo[k,k1,perindex]:=cap1+1;
     end;  { if cap>0}
   end;  {k1=1 to CN}
 end; {k=1 to MCtype}

{Calculation of the Idle time of the machines}
For k:=1 to MCType do
 begin
```

```
   For k1:=1 to CN do
    begin
     Idle[k,k1,perindex]:=(Cap*MacNo[k,k1,perindex])-Capacity[k,k1,perindex];
    end;
  end;
end; {Parameter Calculation}

      {**************PROCEDURE FEASIBILITY***************}
Procedure Feasibility (var feasible : individual;
              var h5:integer);
var
 h,h1,h2,h3,h4,excess,totalop : integer;
begin
 For h:=1 to Operation do
  begin
   totalop:=0;
   For h1:=1 to CN do
    begin
     h2:=h+(operation*(h1-1));
     totalop:=totalop+feasible.chrom[h5,h2];
    end;
   If (totalop=0) then
    begin
     h3:=random(CN)+1;
     h4:=h+(operation*(h3-1));
     feasible.chrom[h5,h4]:=1;
    end;
   If (totalop > 1) then
    begin
     excess:=totalop-1;
     h1:=0;
     Repeat
      h2:=random(CN)+1;
      h3:=h+(operation*(h2-1));
      If (feasible.chrom[h5,h3]=1) then
       begin
        h1:=h1+1;
        feasible.chrom[h5,h3]:=0;
       end;
     Until (excess=h1);
    end;
  end; {h=1 to operation}
end;

      {**************PROCEDURE CROSSOVER***************}
Procedure CrossOver(l:integer;var rf2:integer);
Var
 Mate1, Mate2, DoCrossOver, Crosssite, cr1,rf,rf1,rf3:integer;
begin
 DoCrossOver:=flip(pcrossover);
 Mate1:=random(pop)+1;
 Mate2:=random(pop)+1;

 If (DoCrossOver=1) then
  begin
   ncrossover:=ncrossover+1;
   crosssite:=random(MaxOperation);
   If (crosssite=0) then
    crosssite:=crosssite+1;
  end;
 If (DoCrossOver=0) then
  crosssite:=MaxOperation;

 For cr1:=1 to crosssite do
  begin
   oldpop[l].chrom[rf2,cr1]:=mutation(oldpop[mate1].chrom[rf2,cr1]);
   oldpop[l+1].chrom[rf2,cr1]:=mutation(oldpop[mate2].chrom[rf2,cr1]);
  end;
 If (crosssite <> MaxOperation) then
  begin
   For cr1:=(crosssite+1) to MaxOperation do
    begin
     oldpop[l].chrom[rf2,cr1]:=mutation(oldpop[mate2].chrom[rf2,cr1]);
     oldpop[l+1].chrom[rf2,cr1]:=mutation(oldpop[mate1].chrom[rf2,cr1]);
```

```
    end;
  end;
 rf3:=rf2;
 Feasibility(oldpop[l],rf3);
 Feasibility(oldpop[l+1],rf3);
end;

          {**********PROCEDURE DATA INPUT**************}
{This procedure stores production requirements of parts and all other cost values
into the related arrays for all the periods}
Procedure DataInput;
var
 pd : integer;
begin
 For pd:=1 to Period do
  begin
   Write('Time Data File, Period ',pd,': ');
   Readln(Name);
   Assign(MacFile,Name);
   Reset(MacFile);
   For i:=1 to Part do
    begin
     OpNo[i,pd]:=0;
     For k:=1 to MCType do
      begin
       Read(MacFile,num);
       Time[i,k,pd]:=num;
       If (num<>0) then
        begin
         OpNo[i,pd]:=OpNo[i,pd]+1;
         Test[i,pd]:=k;
        end;
      end;
     Readln(MacFile);
    end;
 Close(MacFile);
 end;

 Write('Enter the Idle time file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
 For i:=1 to MCType do
  begin
   Readln(Macfile,num);
   Idcost[i]:=num;
  end;
 Close(MacFile);

 Write('Enter the Capital Cost file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
 For i:=1 to MCType do
  begin
   Readln(Macfile,num);
   Capital[i]:=num;
  end;
 Close(MacFile);

end;

      {**********PROCEDURE MACHINECHECK**************}
Procedure MachineCheck (var pm3:integer);
Var
 pm,pm1,pm2:integer;

begin
 For pm:=1 to CN do
  begin
   pm2:=0;
   For pm1:=1 to MCType do
    begin
     pm2:=pm2+MacNo[pm1,pm,pm3];
    end;
```

```
    If (pm2 <3) then
     begin
      ok:=false;
     end;
  end;
end;

          {**********PROCEDURE INITDATA**************}
Procedure InitData;
Var
 j, j1,j2,j3,j4,j5,j6,j7 : integer;

begin
 For j:=1 to Pop2 do with oldpop[j] do
  begin
   For t:=1 to Period do
    begin
     Repeat
      ok:=true;
      For j1:=1 to MaxOperation do
       chrom[t,j1]:=0;
      j4:=0;

      For j1:=1 to Part do
       begin
        For j2:=1 to MCType do
         begin
          If (Time[j1,j2,t]>0) then
           begin
            j3:=Random(CN)+1;
            j4:=j4+1;
            For j5:=1 to CN do
             begin
              j6:=j4+(operation*(j5-1));
              If (j5=j3) then
               chrom[t,j6]:=1;
              If (j5<>j3) then
               chrom[t,j6]:=0;
             end;
           end;
         end;
       end;
      Initmachine(t);
      ParameterCalculation(oldpop[j],t);
      MachineCheck(t);
     Until ok;
    end; {For t:=1 to Period}
    For j7:=1 to Period do
     begin
      Initmachine(j7);
      ParameterCalculation(oldpop[j],j7);
     end;
    Runlindo(oldpop[j]);
    Oldpop[j].fitness:=number;
    Writeln('pop ',j,' ',number:10:3);
  end; {for j:=1 to pop}
end;

        {************PROCEDURE ARRAYSORT****************}
Procedure ArraySort;
var
 NoSwap, Pass, First : integer;
 TempRec : Individual;
begin
 Pass:=1;
 Repeat
  Noswap:=0;
  For First:=1 to (Pop2-Pass) do
   begin
    If (oldpop[first].fitness > oldpop[first+1].fitness) then
     begin
      Temprec:=oldpop[first];
      oldpop[first]:=oldpop[first+1];
      oldpop[first+1]:=Temprec;
```

```
        NoSwap:=NoSwap+1;
       end;
     end;
    Pass:=Pass+1;
  Until (Noswap=0);
end;

   {**********PROCEDURE CALCULATE AVERAGE**************}
{This procedure calculates the average fitness value of the population}
Procedure CalculateAverage;
var
ca: integer;
tot: real;
begin
 avg2:=0;
 tot:=0;
 For ca:=1 to Pop do
  tot:=tot+oldpop[ca].fitness;
 avg2:=(tot/pop);
end;

        {************** PROCEDURE REPORT ***************}
Procedure Report;
 begin
  for i:=1 to Pop2 do
   begin
    Writeln(Resultfile,'Final pop: ',i,' ',oldpop[i].fitness:12:3);
    For t:=1 to Period do
     begin
      Initmachine(t);
      ParameterCalculation(oldpop[i],t);
     end;
    Runlindo(oldpop[i]);
    Writeln(resultfile,number:10:1,'/',ns:10:1,'/',ns1:10:1,'/',
    ns2:10:1,'/',ns3:10:1,'/',ns4:10:3);
    For t:=1 to Period do
     For m:=1 to CN do
      For j:=1 to MCType do
       begin
        Write(resultfile,MacNo[j,m,t],',');
         if j=mctype then
          Writeln(resultfile);
       end;
     Writeln(resultfile);
     For t:=1 to Period do
      For m:=1 to CN do
       For j:=1 to MCType do
        Write(resultfile,idle[j,m,t],',');

    Writeln(resultfile);

 For t:=1 to Period do
  begin
   For k:=1 to MaxOperation do
    begin
     Write(resultfile,oldpop[i].chrom[t,k],',');
     if (k=operation) then
      Writeln(resultfile);
    end;
   Writeln(resultfile);
  end;
 end;
 Writeln(resultfile,'mutation: ',nmutation:10:1);
 Writeln(resultfile,'crossover:',ncrossover:10:1);
 Close(Resultfile);
end;
            {**********MAIN PROGRAM***************}
begin
 ClrScr;
 Randomize;
 Assign(Resultfile,'Res1.dat'),
 Rewrite(Resultfile);
 NMutation:=0;
 NCrossOver:=0;
```

```
 gen:=0;
 DataInput;
 Initdata;
 arraysort;
 CalculateStat;
 reproduce;
 arraysort;
clrscr;
Repeat
 Gen:=Gen+1;
 For t:=1 to Period do
  begin
   Repeat
    ok:=true;
    Crossover(pop+1,t);
    Initmachine(t);
    ParameterCalculation(oldpop[pop+1],t);
    MachineCheck(t);
    Initmachine(t);
    ParameterCalculation(oldpop[pop+2],t);
    MachineCheck(t);
   Until ok;
  end;

 For k:=1 to 2 do
  begin
   For t:=1 to Period do
    begin
     Initmachine(t);
     ParameterCalculation(oldpop[pop+k],t);
    end;
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
  end;
 GoToXY (20,5);
 Write('GENERATION ',gen,' Is Complete');
 arraysort;

 if (gen mod 1000) = 0 then
  begin
   calculateaverage;
   Writeln(resultfile,'gen: ',gen,' average: ',avg2:10:3);
   Writeln(resultfile,'First: ',oldpop[1].fitness:10:3,'
   //last',oldpop[pop].fitness:10:3);
  end;

Until (gen=maxgen);
Report;

end. {Main Program}
```

Program GENMOD IV;

```
Uses crt;
 MaxOperation = 48;              {Number of operations * CN}
 Part = 10;                      {Total number of parts}
 Operation = 24;                 {Number of operations}
 MCType = 7;                     {Actual # of Machines Types}
 Pop = 60;                       {Population Size}
 CN = 2;                         {Number of Machine Cells}
 MaxGen =  80000;                {Maximum number of generations w/o
                                  subcontracting}
 FinalGen = 120000;              {Maximum number of generations for the
                                  whole}
 PMutation = 0.0495;             {Probability of Mutation}
 PCrossOver = 0.9989;            {Probability of CrossOver}
 pop2 = pop+2;
 Period = 3;                     {Number of Periods in the Planning
                                  Horizon}
 Cap = 2000;
 maxop = maxoperation+part;
Type
 Subarray = array[1..Part,1..Period] of real;
 Costarray = array[1..MCType] of real;
 Testarray = array[1..Part,1..Period] of integer;
 Timearray = array[1..Part,1..MCType,1..Period] of integer;
 Chromosome = array[1..Period,1..MaxOp] of integer;
 MachineConfiguration = array[1..MCType,1..CN,1..Period] of integer;
 Idlearray = array[1..MCType,1..CN,1..Period] of integer;
 Individual = Record
        chrom : Chromosome;      {The String Structure}
        fitness : real;          {Objective function Value}
        chance : boolean;        {true if less than the average}
        pselect : real;          {prob of selection to next gen.}
       end;
 Population = array[1..Pop2] of Individual;
```

```
Var
 sub: subarray; {Array that stores the subcontracting costs for each part for each
 period}
 num1,de:real;
 ns,ns1,ns2,ns3,ns4,ns5 : real;
 IdCost, Capital, Production : CostArray; {Arrays that contain the cost values for
 each period}
 Test, OpNo : Testarray;
 Time : Timearray;{Array that stores the production requirements of the parts for
 each period}
 Capacity, MacNo : MachineConfiguration; {Array that stores the number of
 machines in each cell for each period}
 Idle : Idlearray;
 MacFile : Text;
 Resultfile, Dummyfile, Infile, Outfile : Text; {Input & Output Files}
 Name : String[15];
 m, t, i, j, k, num : integer;
 Number : Real; {obj value}
 Dummypop, Oldpop : Population;
 Avg, Avg1, Avg2, SumFitness, SumFitness1 : Real;
 Gen : longint;  {Index for number of generations}
 nmutation, ncrossover : real; {Total # of mutation and crossover occured}
 Min, Max : real;
 best,Minimum1, Maximum1 : integer;
 ok:boolean;

    {***************PROCEDURE DECLERATION***************}

      {************PROCEDURE INITMACHINE************}
Procedure InitMachine(var p2:integer);
{This procedure initializes the contents of all the necessary arrays to zero}
Var
 p,p1 : integer;

begin
 For p:=1 to MCType do
  For p1:=1 to CN do
```

```
    begin
     MacNo[p,p1,p2]:=0;
     Idle[p,p1,p2]:=0;
     Capacity[p,p1,p2]:=0;
    end;
end;

      {**************PROCEDURE RUNLINDO***************}
Procedure RunLindo (var RLINDO: individual);

var
 r1,r2,r3,r4,r5,r6:integer;
 move : real;
begin
 number:=0;
 ns:=0;
 ns1:=0;
 ns2:=0;
 ns3:=0;
 ns4:=0;
 ns5:=0;


 {Calculation of the Capital Investment Cost}
 For r6:=1 to Period do
  For r1:=1 to MCType do
   For r2:=1 to CN do
    ns:=ns+(Capital[r1]*MacNo[r1,r2,r6]);

 {Calculation of the Idle Time Cost}
 For r6:=1 to Period do
  For r1:=1 to MCType do
   For r2:=1 to CN do
    ns1:=ns1+(IdCost[r1]*Idle[r1,r2,r6]);

 {Calculation of the Machining Cost}
 For r6:=1 to Period do
 begin
  r3:=0;
  For r1:=1 to Part do
   begin
    For r2:=1 to MCType do
     begin
      If (Time[r1,r2,r6] > 0) then
       begin
        r3:=r3+1;
        For r4:=1 to CN do
         begin
          r5:=r3+(operation*(r4-1));
          ns2:=ns2+(RLINDO.chrom[r6,r5]*(Time[r1,r2,r6]*Production[r2]));
         end;
       end;
     end;
   end;
 end; {r6:=1 to Period}

{Calculation of the InterCellular Movement Cost}
Move:=0.0;
For r6:=1 to Period do
 begin
  For r1:=1 to CN do
   begin
    r4:=0;
    For r2:=1 to Part do
     begin
      r3:=0;
      Repeat
       r3:=r3+1;
       r4:=r4+1;
       r5:=r4+(operation*(r1-1));
       If (RLindo.chrom[r6,r5]-RLindo.chrom[r6,r5+1])=1 then
        Move:=Move+1.0;
      Until (r3)=(OpNo[r2,r6]-1);
      r4:=r4+1;
```

```
      end;
    end;
  end; {r6:=1 to Period}
 ns3:=(2000*Move);

 {Calculation of the Relocation Cost}
 For r6:=1 to (Period-1) do
  For r1:=1 to CN do
   For r2:=1 to MCType do
    begin
     ns4:=ns4+(5000*abs(MacNo[r2,r1,r6+1]-MacNo[r2,r1,r6]));
    end;

 {Calculation of the Subcontracting Cost}
 For r6:=1 to Period do
  For r1:=1 to Part do
   begin
    ns5:=ns5+(Rlindo.chrom[r6,maxoperation+r1]*Sub[r1,r6]);
   end;
 number:=ns+ns1+ns2+ns3+ns4+ns5;
end;

     {************PROCEDURE CALCULATESTAT*************}
Procedure CalculateStat;
Var
 st1, selected :integer;
begin
 sumfitness:=0.0;   avg:=0.0;
 Sumfitness:=oldpop[1].fitness;
 Min:=oldpop[1].fitness;
 Max:=oldpop[1].fitness;
 Maximum1:=1;
 Minimum1:=1;
 For st1:=2 to pop2 do with oldpop[st1] do
  begin
   Sumfitness:=Sumfitness+fitness;
   If fitness>max then
    begin
     max:=fitness;
     maximum1:=st1;
    end;
   If fitness<min then
    begin
     min:=fitness;
     minimum1:=st1;
    end;
  end;
 avg:=(sumfitness/pop2);
 sumfitness1:=0.0;
 avg1:=0.0;
 selected:=0;

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
   If (fitness <= avg) then
    begin
     chance:=true;
     sumfitness1:=sumfitness1+fitness;
     selected := selected + 1;
    end;
   If (fitness > avg) then
    chance:=false;
  end;
 avg1:=(sumfitness1/selected);

 For st1:=1 to pop2 do with oldpop[st1] do
  begin
   If chance then
    pselect:=((2*avg1-fitness)/sumfitness1);
   If not(chance) then
    pselect:=0;
  end;

end; {end of calculatestat}
```

```
{*********FUNCTION FLIP***************}
Function flip(probability:real):integer;
Var
 t1:real;

begin
 If (probability=1) then
  flip:=1;
 If (probability<>1) then
  begin
   t1:=random;
   If (t1<=probability) then
    flip:=1
   else
    flip:=0;
  end;
end;

{************FUNCTION MUTATION***********}
Function Mutation(Mut:integer):integer;
Var
 DoMutation, Dum1 : Integer;
begin
  DoMutation:=flip(Pmutation);
 If (DoMutation=1) then
  begin
   nmutation:=nmutation+1;
   If (Mut=1) then
    Dum1:=0;
   If (Mut=0) then
    Dum1:=1;
  end;
 If (DoMutation=0) then
  Dum1:=Mut;
 Mutation:=Dum1;
end;
```

```
{**************PROCEDURE REPRODUCE*************}
Procedure Reproduce;
Var
 r1,r2,full : integer;
 pselect : real;
begin
 full:=0;
 r1:=0;
 Repeat
  r1:=r1+1;
  If (oldpop[r1].chance) then
   r2:=flip(oldpop[r1].pselect);
  If (r2=1) and (oldpop[r1].chance) then
   begin
    full:=full+1;
    dummypop[full]:=oldpop[r1];
   end;
  If (r1=pop2) then
   r1:=0;
 Until (full=pop2);
 oldpop:=dummypop;
end;

{*************PROCEDURE PARAMETERCALCULATION*********}
Procedure ParameterCalculation (var paracal : individual;
                  var perindex:integer);

Var
 k,k1,k2,k3,k4,k5,cap1 : integer;

begin
 k2:=0;
 For k:=1 to Part do
  begin
   For k1:=1 to MCType do
    begin
     If (Time[k,k1,perindex]) > 0 then
```

```
      begin
       k2:=k2+1;
       For k3:=1 to CN do
        begin
         k4:=k2+(operation*(k3-1));
         Capacity[k1,k3,perindex]:=Capacity[k1,k3,perindex]+
         (Time[k,k1,perindex]*paracal.chrom[perindex,k4]);
        end;
      end; {if time[k,k1]>0}
    end; {for k1:=1 to mctype}
 end;  {for k:=1 to part}

{Calculation of the number of machines in each cell for each period}
For k:=1 to MCType do
 begin
  For k1:=1 to CN do
   begin
    If (Capacity[k,k1,perindex]=0) then
     MacNo[k,k1,perindex]:=0;
    If (Capacity[k,k1,perindex]>0) then
     begin
      Cap1:=Capacity[k,k1,perindex] div cap;
      If (Capacity[k,k1,perindex] mod cap = 0) then
       MacNo[k,k1,perindex]:=cap1;
      If (Capacity[k,k1,perindex] mod cap > 0) then
       MacNo[k,k1,perindex]:=cap1+1;
     end;  { if cap>0}
   end;  {k1=1 to CN}
 end; {k=1 to MCtype}

{Calculation of the idle time of machines for each period in each machine cell}
For k:=1 to MCType do
 begin
  For k1:=1 to CN do
   begin
    Idle[k,k1,perindex]:=(Cap*MacNo[k,k1,perindex])-Capacity[k,k1,perindex];
   end;
  end;
end; {Parameter Calculation}

        {**********PROCEDURE FEASIBILITY**********}
Procedure Feasibility (var feasible : individual;
            var h6:integer);
var
 h,h1,h2,h3,h4,h5,excess,totalop : integer;
begin
 For h:=1 to Operation do
  begin
   totalop:=0;
   For h1:=1 to CN do
    begin
     h2:=h+(operation*(h1-1));
     totalop:=totalop+feasible.chrom[h6,h2];
    end;
   If (totalop=0) then
    begin
     h3:=random(CN)+1;
     h4:=h+(operation*(h3-1));
     feasible.chrom[h6,h4]:=1;
    end;
   If (totalop > 1) then
    begin
     excess:=totalop-1;
     h1:=0;
     Repeat
      h2:=random(CN)+1;
      h3:=h+(operation*(h2-1));
      If (feasible.chrom[h6,h3]=1) then
       begin
        h1:=h1+1;
        feasible.chrom[h6,h3]:=0;
       end;
      Until (excess=h1);
    end;
```

```
 end; {h=1 to operation}

For h:=1 to Part do
 begin
  If (feasible.chrom[h6,maxoperation+h]=1) then
   begin
    h2:=0;
    For h1:=1 to (h-1) do
     h2:=h2+OpNo[h1,h6];
    For h3:=1 to CN do
     begin
      For h4:=1 to OpNo[h,h6] do
       begin
        h5:=h2+h4+(operation*(h3-1));
        feasible.chrom[h6,h5]:=0;
       end;
     end;
   end;
 end;   {for h:=1 to Part}

end; {feasibility}


      {**********PROCEDURE CROSSOVER**********}
Procedure CrossOver(l:integer;var rf2:integer);
Var
 Mate1, Mate2, DoCrossOver, Crosssite, cr1,rf,rf1,rf3:integer;
begin
 DoCrossOver:=flip(pcrossover);
 Mate1:=random(Pop)+1;
 Mate2:=random(Pop)+1;

 If (DoCrossOver=1) then
  begin
   ncrossover:=ncrossover+1;
   crosssite:=random(MaxOperation);
   If (crosssite=0) then
    crosssite:=crosssite+1;
  end;
 If (DoCrossOver=0) then
  crosssite:=MaxOperation;

 For cr1:=1 to crosssite do
  begin
   oldpop[l].chrom[rf2,cr1]:=mutation(oldpop[mate1].chrom[rf2,cr1]);
   oldpop[l+1].chrom[rf2,cr1].=mutation(oldpop[mate2].chrom[rf2,cr1]);
  end;
 If (crosssite <> MaxOperation) then
  begin
   For cr1:=(crosssite+1) to MaxOperation do
    begin
     oldpop[l].chrom[rf2,cr1]:=mutation(oldpop[mate2].chrom[rf2,cr1]);
     oldpop[l+1].chrom[rf2,cr1]:=mutation(oldpop[mate1].chrom[rf2,cr1]);
    end;
  end;
 For cr1:=1 to Part do
  begin
   oldpop[l].chrom[rf2,maxoperation+cr1]:=oldpop[mate1].
   chrom[rf2,maxoperation+cr1];
   oldpop[l+1].chrom[rf2,maxoperation+cr1]:=oldpop[mate2].
   chrom[rf2,maxoperation+cr1];
  end;
 rf3:=rf2;
 Feasibility(oldpop[l],rf3);
 Feasibility(oldpop[l+1],rf3);
end;

          {**********PROCEDURE DATA INPUT**********}
{This procedure stores production requirements of the parts and all other cost
values into related arrays for each period in the planning horizon}

Procedure DataInput;
var
 pd:integer;
```

```
begin
 For pd:=1 to Period do
  begin
   Write('Time Data File, Period ',pd,' : ');
   Readln(Name);
   Assign(MacFile,Name);
   Reset(MacFile);
   For i:=1 to Part do
    begin
     OpNo[i,pd]:=0;
     For k:=1 to MCType do
      begin
       Read(MacFile,num);
       Time[i,k,pd]:=num;
       If (num<>0) then
        begin
         OpNo[i,pd]:=OpNo[i,pd]+1;
         Test[i,pd]:=k;
        end;
      end;
     Readln(MacFile);
    end;
   Close(MacFile);
  end;

 Write('Enter the Idle time file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
 For i:=1 to MCType do
  begin
   Readln(Macfile,num1);
   Idcost[i]:=num1;
  end;
 Close(MacFile);

 Write('Enter the Capital Cost file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
 For i:=1 to MCType do
  begin
   Readln(Macfile,num1);
   Capital[i]:=num1;
  end;
 Close(MacFile);


 Write('Enter the Production Cost file==> ');
 Readln(Name);
 Assign(Macfile,Name);
 Reset(MacFile);
 For i:=1 to MCType do
  begin
   Readln(Macfile,num1);
   Production[i]:=num1;
  end;
 Close(MacFile);

 For pd:=1 to Period do
  begin
   Write('Enter the Subcontracting Cost File, Period ',pd,' : ');
   Readln(Name);
   Assign(MacFile,Name);
   Reset(MacFile);
   For i:=1 to Part do
    begin
     Readln(MacFile,num1);
     Sub[i,pd]:=num1;
    end;
   Close(MacFile);
  end;
end;
```

```
      {**********PROCEDURE MACHINECHECK***********}
Procedure MachineCheck(var pm3:integer);

Var
 pm,pm1,pm2:integer;

begin
  For pm:=1 to CN do
   begin
    pm2:=0;
    For pm1:=1 to MCType do
     begin
      pm2:=pm2+MacNo[pm1,pm,pm3];
     end;
     If (pm2 <3) then
      begin
       ok:=false;
      end;
   end;
end;

        {************ PROCEDURE FINDTHEBEST**********}
Procedure FindTheBest (var subrecord:individual; var s7:integer);
var
 srecord : individual;
 s1,s2,s3,s4,s5,s6,s8,pf,pf1 : integer;
 num2,num3:real;
begin
 num2:=subrecord.fitness;
 num3:=subrecord.fitness;
 For s1:=1 to Part do
  begin
   srecord:=subrecord;
   ok:=true;
   s3:=0;
   srecord.chrom[s7,maxoperation+s1]:=1;
   For s2:=1 to (s1-1) do
    s3:=s3+OpNo[s2,s7];
   For s4:=1 to CN do
    begin
     For s5:=1 to OpNo[s1,s7] do
      begin
       s6:=s3+s5+(operation*(s4-1));
       srecord.chrom[s7,s6]:=0;
      end;
    end;

   For s8:=1 to Period do
    begin
     Initmachine(s8);
     ParameterCalculation(srecord,s8);
    end;
   machinecheck(s7);
   If ok then
    begin
     Runlindo(srecord);
     If (number < num2) and (number < num3) then
      begin
       best:=s1;
       num3:=number;
      end;
    end;
   srecord.chrom[s7,maxoperation+s1]:=0;
  end;
end;

     {*********PROCEDURE SUBCONTRACT THE PART********}
Procedure SubcontractThePart (var stprecord:individual;var st5:integer);
var
 st1,st2,st3,st4 : integer;

begin
 st1:=0;
 For st2:=1 to (best-1) do
```

```
  st1:=st1+OpNo[st2,st5];
 For st2:=1 to CN do
  begin
   For st3:=1 to OpNo[best,st5] do
    begin
     st4:=st1+st3+(operation*(st2-1));
     stprecord.chrom[st5,st4]:=0;
    end;
  end;
 stprecord.chrom[st5,maxoperation+best]:=1;
end;

          {*************PROCEDURE INITDATA*************}
Procedure InitData;
Var
 j, j1,j2,j3,j4,j5,j6,j7 : integer;

begin
 For j:=1 to Pop2 do with oldpop[j] do
  begin
   For t:=1 to Period do
    begin
     Repeat
      ok:=true;
      For j1:=1 to (MaxOp) do
       chrom[t,j1]:=0;
      j4:=0;

      For j1:=1 to Part do
       begin
        For j2:=1 to MCType do
         begin
          If (Time[j1,j2,t]>0) then
           begin
            j3:=Random(CN)+1;
            j4:=j4+1;
            For j5:=1 to CN do
                begin
                 j6:=j4+(operation*(j5-1));
                 If (j5=j3) then
                  chrom[t,j6]:=1;
                 If (j5<>j3) then
                  chrom[t,j6]:=0;
                end;
           end;
         end;
       end;
      Initmachine(t);
      ParameterCalculation(oldpop[j],t);
      MachineCheck(t);
     Until ok;
    end; {For t:=1 to Period}
    For j7:=1 to Period do
     begin
      Initmachine(j7);
      ParameterCalculation(oldpop[j],j7);
     end;
    Runlindo(oldpop[j]);
    Oldpop[j].fitness:=number;
    Writeln('pop ',j,' ',number:10:3);
  end; {for j:=1 to pop}
end;

          {**********PROCEDURE ARRAYSORT**********}
Procedure ArraySort;
var
 NoSwap, Pass, First : integer;
 TempRec : Individual;
begin
 Pass:=1;
 Repeat
  Noswap:=0;
  For First:=1 to (Pop2-Pass) do
   begin
```

```
    If (oldpop[first].fitness > oldpop[first+1].fitness) then
     begin
      Temprec:=oldpop[first];
      oldpop[first]:=oldpop[first+1];
      oldpop[first+1]:=Temprec;
      NoSwap:=NoSwap+1;
     end;
   end;
  Pass:=Pass+1;
 Until (Noswap=0);

end;

    {**********PROCEDURE CALCULATE AVERAGE**********}
{This procedure calculates the average fitness value of the population}

Procedure CalculateAverage;
var
 ca: integer;
 tot: real;
begin
 avg2:=0;
 tot:=0;
 For ca:=1 to Pop do
  tot:=tot+oldpop[ca].fitness;
 avg2:=(tot/pop);
end;

          {********** PROCEDURE REPORT **********}
Procedure Report;
begin
 For i:=1 to Pop2 do
  begin
   Writeln(Resultfile,'Final pop: ',i,' ',oldpop[i].fitness:12:3);
    For t:=1 to Period do
     begin
      Initmachine(t);
      ParameterCalculation(oldpop[i],t);
     end;

 Runlindo(oldpop[i]);
 Writeln(resultfile,number:10:1,'/',ns:10:1,'/',ns1:10:1,'/',ns2:10:1,'/',
 ns3:10:1,'/',ns4:10:1,'/',ns5:10:3);
 For t:=1 to period do
  For m:=1 to CN do
   For j:=1 to MCType do
    begin
     Write(resultfile,MacNo[j,m,t],',');
     If j=mctype then
      Writeln(resultfile);
    end;
 Writeln(resultfile);

 For t:=1 to Period do
  For m:=1 to CN do
   For j:=1 to MCType do
    Write(resultfile,idle[j,m,t],',');
 Writeln(resultfile);

 For t:=1 to Period do
  begin
   For k:=1 to MaxOp do
    begin
     Write(resultfile,oldpop[i].chrom[t,k],',');
     If (k = operation) then Writeln(resultfile);
    end;
  end;
 Writeln(Resultfile);
end;

Writeln(resultfile,'mutation: ',nmutation:10:1);
Writeln(resultfile,'crossover:',ncrossover:10:1);
Close(Resultfile);
end;
```

```
{*************MAIN PROGRAM***************}

begin
 ClrScr;
 Randomize;
 Assign(Resultfile,'Res1.dat');
 Rewrite(Resultfile);
 NMutation:=0;
 NCrossOver:=0;
 gen:=0;
 DataInput;
 Initdata;
 arraysort;
 calculatestat;
 reproduce;
 arraysort;
 clrscr;
 Repeat
  Gen:=Gen+1;
  For t:=1 to Period do
   begin
    Repeat
     ok:=true;
     Crossover(pop+1,t);
     Initmachine(t);
     ParameterCalculation(oldpop[pop+1],t);
     MachineCheck(t);
     Initmachine(t);
     ParameterCalculation(oldpop[pop+2],t);
     MachineCheck(t);
    Until ok;
   end;

 For k:=1 to 2 do
  begin
   For t:=1 to Period do
    begin
     Initmachine(t);
     ParameterCalculation(oldpop[pop+k],t);
    end;
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
  end;
 gotoxy(15,4);
 Write('GENERATION ',gen,' Is Complete ');
 arraysort;
 if (gen mod 1000) = 0 then
  begin
   calculateaverage;
   Writeln(resultfile,'gen: ',gen,' average: ',avg2:10:3);
   Writeln(resultfile,'First: ',oldpop[1].fitness:10:3,
   ' //Last',oldpop[pop].fitness:10:3);
  end;
Until (gen=maxgen);

{Subcontracting Begins}
For i:=1 to Pop do
 begin
  For t:=1 to Period do
   begin
    Repeat
     best:=part+1;
     FindTheBest(oldpop[i],t);
     If((Best<>(Part+1))and (oldpop[i].chrom[t,maxoperation+Best]<>1))then
      begin
       SubcontractThePart(oldpop[i],t);
       For m:=1 to Period do
        begin
         Initmachine(t);
         ParameterCalculation(oldpop[i],t);
        end;
       RunLindo(oldpop[i]);
```

```
        oldpop[i].fitness:=number;
       end;
     Until (Best=part+1);
    end;
end;
arraysort;
Writeln(resultfile);

Repeat
 Gen:=Gen+1;
 For t:=1 to Period do
  begin
   Repeat
    ok:=true;
    Crossover(pop+1,t);
    Initmachine(t);
    ParameterCalculation(oldpop[pop+1],t);
    MachineCheck(t);
    Initmachine(t);
    ParameterCalculation(oldpop[pop+2],t);
    MachineCheck(t);
   Until ok;
  end;

 For k:=1 to 2 do
  begin
   For t:=1 to Period do
    begin
     Initmachine(t);
     ParameterCalculation(oldpop[pop+k],t);
    end;
   Runlindo(oldpop[pop+k]);
   oldpop[pop+k].fitness:=number;
  end;
 gotoxy(15,4);
 Writeln('GENERATION ',gen,' Is Complete');
 arraysort;
 if (gen mod 1000) = 0 then
  begin
   calculateaverage;
   Writeln(resultfile,'gen: ',gen,' average: ',avg2:10:3);
   Writeln(resultfile,'First: ',oldpop[1].fitness:10:3,
   ' //Last',oldpop[pop].fitness:10:3);
  end;
Until (gen=finalgen);
Report;
end. {Main Program}
```

APPENDIX II

Source Code of the Input File Generators

```
Program InputCreator {SINGLE PERIOD};

Uses Crt;

Const
 Part = 20;          {Number of part types}
 Mac = 12;           {Number of machine types}
 CN = 2;             {Number of machine cells to be created}
 MAX = 10;           {Maximum number of machines allowed in the cell}
 MIN = 5;            {Minimum number of machines allowed in the cell}
 InterCell = 2000;   {Unit Intercellular Movement Cost}
 Capacity = 2000;    {Unit capacity of a machine type}

label 100;

Type
 MultiArray = array[1..Part] of integer;

 TimeArray = array[1..Part,1..Mac] of integer; {Array that stores the machine
 times of the parts, the input data should be entered as the multiplication of the
 demand and the processing time}

 CapitalCost = array[1..Mac] of LongInt; {Array that stores the unit capital
 investment for each machine type}

 MachineCost = array[1..Mac] of Real; {Array that stores the unit machining cost
 for each machine type}


 IdleTimeCost = array[1..Mac] of Real; {Array that stores the idle time cost for
 each machine type}

 SubCostArray = array[1..Part] of Longint; {Array that stores the subcontracting
 cost for each part type for the whole demand}

Var
 Found, Stop, Quit, Sub : Boolean;
 First, Second, Num, i, j, k : Integer;
 Number : longint;
 Num1  : Real;
 Time  : TimeArray;
 Test, OpNum : MultiArray;
 SubCost : SubCostArray;
 Capital : CapitalCost;
 MCost : MachineCost;
 Idle : IdleTimeCost;
 Infile, Infile1, Outfile : Text;
 Ch : char;
 Name : String[12];

Begin
 Clrscr;
 Write('Subcontracting Allowed? (Y/N):  ');
 Readln(Ch);
 Ch:=UpCase(Ch);
 If (Ch='Y') then
  Sub:=True
 Else
  Sub:=False;

 Write('Enter the Time.dat:  ');
 Readln(Name);
 Assign(Infile,Name);
 Reset(Infile);

 Write('Enter the Output file :  ');
 Readln(Name);
 Assign(Outfile,Name);
 Rewrite(Outfile);

 {Input the Unit Capital Investment Cost for Each Machine Type}
 Write('Enter the Capital Cost Matrix : ');
```

```
Readln(Name);
Assign(Infile1,Name);
Reset(Infile1);
For k:=1 to Mac do
 begin
  Readln(Infile1,Number);
  Capital[k] := Number;
 end;
Close(Infile1);

{Input the Unit Machining Cost ($/hr) for Each Machine Type}
If (Sub =True) then
 begin
  Write('Enter the Machine Cost Matrix : ');
  Readln(Name);
  Assign(Infile1,Name);
  Reset(Infile1);
  For k:=1 to Mac do
   begin
    Readln(Infile1,Num1);
    MCost[k] := Num1;
   end;
  Close(Infile1);
 end;

{Input Idle Time Cost ($/hr) for Each Machine Type}
Write('Enter the Idle Time Cost Matrix : ');
Readln(Name);
Assign(Infile1,Name);
Reset(Infile1);
For k:=1 to Mac do
 begin
  Readln(Infile1,Num1);
  Idle[k] := Num1;
 end;
Close(Infile1);

{If Subcontracting Is Allowed, Input the Subcontracting
 Cost for Each Part Type, (Unit Cost * Demand)}
If (Sub=true) then
 begin
  Write('Enter the Subcontracting Cost Matrix:  ');
  Readln(Name);
  Assign(Infile1,Name);
  Reset(Infile1);
  For i:=1 to Part do
   begin
    Readln(Infile1,Number);
    SubCost[i]:=Number;
    OpNum[i]:=0;
   end;
  Close(Infile1);
 end;

{Read the Time input data and the store the index of the
 last operation for testing purposes}
For i:=1 to Part do
 begin
  For k:=1 to Mac do
   begin
    Read(Infile,Number);
    Time[i,k] := Number;
    If (Number <> 0) then
     begin
      Test[i] := k;
      OpNum[i]:=OpNum[i]+1;
     end;
   end;
   Readln(Infile);
 end;
Close(Infile);

{The Output File to Specify the Binary and General Integer Variables}
```

```
Assign(Infile1,'Var.dat');
Rewrite(Infile1);

{Objective Function Development}
{Capital Invesment Cost Component}
Writeln(Outfile,'!Machine Investment Cost');
For j:=1 to CN do
 begin
  For k:=1 to Mac do
   Write(Outfile,Capital[k],'Y',k,'_',j,' + ');
  end;
Writeln(Outfile,'!');
Writeln(Outfile,'!');

{Intercell Movement Cost Component}
For j:=1 to CN do
 begin
  For i:=1 to Part do
   begin
    For k:=1 to MAC do
     begin
      If ((Time[i,k] <> 0) and (Test[i] <> k)) then
       begin
        Write(Outfile,InterCell,'M',i,'_',k,'_',j,'+ ');
        Writeln(Infile1,'INT M',i,'_',k,'_',j);
       end;
     end;
    Writeln(Outfile);
   end;
  Writeln(outfile);
 end;

{Idle Time Cost Component}
For j:=1 to CN do
 begin
  For k:=1 to Mac do
   Write(Outfile,Idle[k]:3:1,'ID',k,'_',j,' + ');
 end;
Writeln(Outfile,'!');
Writeln(Outfile,'!');

If (Sub=True) Then
begin
{Machining Cost Component}
For j:= 1 to CN do
 begin
  For i:=1 to Part do
   begin
    For k:=1 to MAC do
     begin
      If not (Time[i,k]=0) then
       begin
        num1:=time[i,k]*Mcost[k];
        Write(Outfile,Num1:5:1,'Z',i,'_',k,'_',j,' + ');
       end;
     end;
    Writeln(Outfile);
   end;
 end;
 end;

{Subcontracting Cost Component}
If (Sub=True) then
begin
 Writeln(Outfile,'!SubContracting Cost');
 For i:=1 to Part do
  begin
   Write(Outfile,SubCost[i],'S',i,' + ');
   Writeln(Infile1,'INT S',i);
  end;
 Writeln(Outfile);
end;
```

```
{Constraint Set Development}

{Operation Allocation}
For i:=1 to Part do
 begin
  For k:=1 to Mac do
   begin
    If not (Time[i,k]=0) then
     begin
      For j:=1 to (CN-1) do
       begin
        Write(Outfile,'Z',i,'_',k,'_',j,' + ');
        Writeln(Infile1,'INT Z',i,'_',k,'_',j);
       end;
      If (Sub <> true) then
       Writeln(Outfile,'Z',i,'_',k,'_',CN,' = 1');
      If (Sub = true) then
       Writeln(Outfile,'Z',i,'_',k,'_',CN,' <= 1');
      Writeln(Infile1,'INT Z',i,'_',k,'_',CN);
     end;
   end;
 end;

{Subcontracting Constraint}
If (Sub=true) then
 begin
  Writeln(Outfile,'! Subcontracting Constraint');
   For i:=1 to Part do
    begin
     For j:=1 to CN do
      begin
       For k:=1 to MAC do
        begin
         If (Time[i,k] <> 0) then
          Write(Outfile,'Z',i,'_',k,'_',j,' + ');
        end;
       end;
      Writeln(Outfile,OpNum[i],'S',i,' = ',OpNum[i]);
    end;
   end;

{Limitation on the number of machines in each cell}
For j:=1 to CN do
 begin
  For k:=1 to (MAC-1) do
   begin
    Write(Outfile,'Y',k,'_',j,' + ');
   end;
  Writeln(Outfile,'Y',k+1,'_',j,' >=',MIN);
 end;

 For j:=1 to CN do
 begin
  For k:=1 to (MAC-1) do
   begin
    Write(Outfile,'Y',k,'_',j,' + ');
   end;
  Writeln(Outfile,'Y',k+1,'_',j,' >=',MAX);
 end;

{Capacity Requirements}
Writeln(Outfile);
Writeln(Outfile,'! Constraint 8');
For j:=1 to CN do
 begin
  For k:=1 to MAC do
   begin
    For i:=1 to Part do
     begin
      If not (Time[i,k] = 0) then
       begin
        Write(Outfile,Time[i,k],'Z',i,'_',k,'_',j,' + ');
```

```
        end;
      end;
     Writeln(Outfile,'ID',k,'_',j,' - ',Capacity,'Y',k,'_',j,' = 0');
     Writeln(Infile1,'GIN Y',k,'_',j);
    end;
  end;

 {Intercellular Movements}
 For j:=1 to CN do
  begin
   For i:=1 to Part do
    begin
     k:=0;
     Found:= False;
     Repeat
      k:=k+1;
      If not (Time[i,k]=0) then
       begin
        Found := True;
        First := k;
       end;
     Until Found;
100:   Stop := False;
     Repeat
      k:=k+1;
      If not(Time[i,k]=0) then
       begin
        Second:=k;
        Stop := True;
       end;
      Until (Stop or (k=Mac));
      If (k<>mac) or ((Time[i,first]>0)and (Time[i,k]>0))  then
      Writeln(Outfile,'Z',i,'_',First,'_',j,' - ','Z',i,'_',Second,
      '_',j,' - ','M',i,'_',First,'_',j,' <= 0');
      First:=Second;
      Quit :=False;
      If (k=Mac-1) and (Time[i,k]=0) then
       Quit:=true;
      If (not (k=Mac)) and not(Quit) then
       Goto 100;
    end;
  end;

 Close(Infile1);
 Close(Outfile);
End. {Main Program}
```

```
Program InputCreator {MULTI PERIOD};

Uses Crt;

Const
 Part = 10;          {Number of part types}
 Mac = 7;            {Number of machine types}
 CN = 2;             {Number of machine cells to be created}
 MAX = 5;            {Maximum number of machines allowed in the cell}
 MIN = 3;            {Minimum number of machines allowed in the cell}
 Period = 3;         {Number of Periods}
 InterCell = 2000;   {Unit Intercellular Movement Cost}
 Capacity = 2000;    {Unit capacity of a machine type}
 Location = 5000;    {Unit relocation cost per machine}

label 100;

Type
 MultiArray = array[1..Part,1..Period] of integer;

 TimeArray = array[1..Part,1..Mac,1..Period] of integer; {Array that stores the
 machine times of the parts, the input data should be entered as the multiplication
 of the demand and the processing time}

 CapitalCost = array[1..Mac] of LongInt; {Array that stores the unit capital
 investment for each machine type}

 MachineCost = array[1..Mac] of Real; {Array that stores the unit machining cost
 for each machine type}

 IdleTimeCost = array[1..Mac] of Real; {Array that stores the idle time cost for
 each machine type}

 SubCostArray = array[1..Part,1..Period] of Longint; {Array that stores the
 subcontracting cost for each part type for the whole demand}

Var
 Found, Stop, Quit, Sub : Boolean;
 First, Second, Num, i, j, k, t : Integer;
 Number : longint;
 Num1  : Real;
 Time  : TimeArray;
 Test, OpNum : MultiArray;
 SubCost : SubCostArray;
 Capital : CapitalCost;
 MCost : MachineCost;
 Idle : IdleTimeCost;
 Infile, Infile1, Outfile : Text;
 Ch : char;
 Name : String[12];

Begin
 Clrscr;
 Write('Subcontracting Allowed? (Y/N): ');
 Readln(Ch);
 Ch:=UpCase(Ch);

 If (Ch='Y') then
  Sub:=True
 Else
  Sub:=False;

 Write('Enter the Output file :  ');
 Readln(Name);
 Assign(Outfile,Name);
 Rewrite(Outfile);

 {Input the Unit Capital Investment Cost for Each Machine Type}
 Write('Enter the Capital Cost Matrix : ');
 Readln(Name);
 Assign(Infile1,Name);
 Reset(Infile1);
 For k:=1 to Mac do
```

```
 begin
  Readln(Infile1,Number);
  Capital[k] := Number;
 end;
Close(Infile1);

{Input the Unit Machining Cost ($/hr) for Each Machine Type}
Write('Enter the Machine Cost Matrix : ');
Readln(Name);
Assign(Infile1,Name);
Reset(Infile1);
For k:=1 to Mac do
 begin
  Readln(Infile1,Num1);
  MCost[k] := Num1;
 end;
Close(Infile1);

{Input Idle Time Cost ($/hr) for Each Machine Type}
Write('Enter the Idle Time Cost Matrix : ');
Readln(Name);
Assign(Infile1,Name);
Reset(Infile1);
For k:=1 to Mac do
 begin
  Readln(Infile1,Num1);
  Idle[k] := Num1;
 end;
Close(Infile1);

{If Subcontracting Is Allowed, Input the Subcontracting Cost for Each Part
 Type for each Period (Unit Cost * Demand)}
If (Sub=true) then
 begin
  For t:=1 to Period do
  begin
  Write('Enter the Subcontracting Cost Matrix, Period ',t,':  ');
  Readln(Name);
  Assign(Infile1,Name);
  Reset(Infile1);
  For i:=1 to Part do
   begin
    Readln(Infile1,Number);
    SubCost[i,t]:=Number;
   end;
  Close(Infile1);
  end;
 end;

For t:=1 to Period do
 begin
  For i:=1 to Part do
   OpNum[i,t]:=0;
 end;

{Read the Time input data for each period and the store the index of the
 last operation for testing purposes}
For t:=1 to Period do
begin
Write('Enter the Demand Data for Period ',t,' :');
Readln(Name);
Assign(Infile,Name);
Reset(Infile);
For i:=1 to Part do
 begin
  For k:=1 to Mac do
   begin
    Read(Infile,Number);
    Time[i,k,t] := Number;
    If (Number <> 0) then
     begin
      Test[i,t] := k;
      OpNum[i,t]:=OpNum[i,t] + 1;
     end;
```

```
    end;
    Readln(Infile);
  end;
Close(Infile);
end;
{The Output File to Specify the Binary and General Integer Variables}
Assign(Infile1,'Var.dat');
Rewrite(Infile1);

{Objective Function Development}

{Capital Investment Cost Component}
Writeln(Outfile,'!Machine Investment Cost');
For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  Writeln(Outfile);
  For k:=1 to Mac do
   Write(Outfile,Capital[k],'Y',k,'_',j,'_',t,' + ');
 end;
end;
Writeln(Outfile,'!');
Writeln(Outfile,'!');

{Intercellular Movement Cost Component}
For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  For i:=1 to Part do
   begin
    For k:=1 to MAC do
     begin
      If ((Time[i,k,t] <> 0) and (Test[i,t] <> k)) then
       begin
        Write(Outfile,InterCell,'M',i,'_',k,'_',j,t,' + ');
        Writeln(Infile1,'INT M',i,'_',k,'_',j,t);
       end;
     end;
    Writeln(Outfile);
   end;
  Writeln(outfile);
 end;
 End;

{Idle Time Cost Component}
For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  For k:=1 to Mac do
   Write(Outfile,Idle[k]:3:1,'ID',k,'_',j,t,' + ');
 end;
end;
Writeln(Outfile,'!');
Writeln(Outfile,'!');

{Machining Cost Component}
For t:=1 to Period do
begin
For j:= 1 to CN  do
 begin
  For i:=1 to Part do
   begin
    For k:=1 to MAC do
     begin
      If not (Time[i,k,t]=0) then
       begin
        num1:=time[i,k,t]*Mcost[k];
        Write(Outfile,Num1:5:1,'Z',i,'_',k,'_',j,t,' + ');
       end;
     end;
    Writeln(Outfile);
```

```
   end;
 end;
 end;

{Subcontracting Cost Component}
If (Sub=True) then
begin
 Writeln(Outfile,'!Subcontracting Cost');
 For t:=1 to Period do
 begin
  For i:=1 to Part do
   begin
    Write(Outfile,SubCost[i,t],'S',i,'_',t,' + ');
    Writeln(Infile1,'INT S',i,'_',t);
   end;
  Writeln(Outfile);
 end;
end;

{Relocation Cost Component}
Writeln(Outfile,'!Relocation cost');
For t:=1 to (Period-1) do
 begin
  For j:=1 to CN do
   begin
    For k:=1 to Mac do
     begin
      Write(Outfile,Location,'PC',k,'_'j,'_',t,' + ',Location,'NC',k,'_',
      j,'_',t,' + ');
     end;
    Writeln(Outfile);
   end;
  Writeln(Outfile);
 end;

{Constraint Development}
```



```
{Operation Allocation}
For t:=1 to Period do
begin
For i:=1 to Part do
 begin
  For k:=1 to Mac do
   begin
    If not (Time[i,k,t]=0) then
     begin
      For j:=1 to (CN-1) do
       begin
        Write(Outfile,'Z',i,'_',k,'_'j,t,' + ');
        Writeln(Infile1,'INT Z',i,'_',k,'_'j,t);
       end;
      If (Sub <> true) then
       Writeln(Outfile,'Z',i,'_',k,'_',CN,t,' = 1');
      If (Sub = true) then
       Writeln(Outfile,'Z',i,'_',k,'_',CN,t,' <= 1');
      Writeln(Infile1,'INT Z',i,'_',k,'_',CN,t);
     end;
   end;
 end;
 end;

{Subcontracting Constraint}
If (Sub=true) then
 begin
  Writeln(Outfile,'!Subcontracting Constraint');
  For t:=1 to Period do
  begin
   For i:=1 to Part do
    begin
     For j:=1 to CN do
      begin
       For k:=1 to MAC do
        begin
         If (Time[i,k,t] <> 0) then
```

```
            Write(Outfile,'Z',i,'_',k,'_',j,t,' + ');
          end;
        end;
       Writeln(Outfile,OpNum[i,t],'S',i,'_',t,' = ',OpNum[i,t]);
     end;
    end;
  end;
Writeln(Outfile);

{Limitation on the number of machines in each cell}
For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  For k:=1 to (MAC-1) do
   begin
    Write(Outfile,'Y',k,'_',j,'_',t,' + ');
   end;
  Writeln(Outfile,'Y',k+1,'_',j,'_',t,' >=',MIN);
 end;
 end;

 For t:=1 to Period do
 begin
 For j:=1 to CN do
 begin
  For k:=1 to (MAC-1) do
   begin
    Write(Outfile,'Y',k,'_',j,'_',t,' + ');
   end;
  Writeln(Outfile,'Y',k+1,'_',j,'_',t,' <=',MAX);
 end;
 end;

{Capacity Requirements}
Writeln(Outfile);

For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  For k:=1 to MAC do
   begin
    For i:=1 to Part do
     begin
      If not (Time[i,k,t] = 0) then
       begin
        Write(Outfile,Time[i,k,t],'Z',i,'_',k,'_',j,t,' + ');
       end;
     end;
    Writeln(Outfile,'ID',k,'_',j,t,' - ',Capacity,'Y',k,'_',j,'_',t,' = 0');
    Writeln(Infile1,'GINY',k,'_',j,'_',t);
   end;
 end;
 end;
Writeln(Outfile,'!');

{Intercellular Movements}
For t:=1 to Period do
begin
For j:=1 to CN do
 begin
  For i:=1 to Part do
   begin
    k:=0;
    Found:= False;
    Repeat
     k:=k+1;
     If not (Time[i,k,t]=0) then
      begin
       Found := True;
       First := k;
      end;
    Until Found;
```

```
100:    Stop := False;
       Repeat
        k:=k+1;
        If not(Time[i,k,t]=0) then
         begin
          Second:=k;
          Stop := True;
         end;
       Until (Stop or (k=Mac));
       If (k<>mac) or ((Time[i,first,t]>0) and (Time[i,k,t]>0))  then
       Writeln(Outfile,'Z',i,'_',First,'_',j,t,' - ','Z',i,'_',Second,'_',j,t,' - ','M',i,'_',
       First,'_',j,t,' <= 0');
       First:=Second;
       Quit :=False;
       If (k=Mac-1) and (Time[i,k,t]=0) then
        Quit:=true;
       If (not (k=Mac)) and not(Quit) then
        Goto 100;
     end;
   end;
   end;

  Writeln(Outfile,'!');
  For t:=1 to (Period-1) do
   begin
    For j:=1 to CN do
     begin
      For k:=1 to MAC do
       begin
        Writeln(Outfile,'Y',k,'_',j,'_',t+1,' - ','Y',k,'_',j,'_',t,' - '
        ,'PC',k,'_',j,'_',t,' + ','NC',k,'_',j,'_',t,' = 0');
        Writeln(Infile1,'INT NC',k,'_',j,'_',t);
        Writeln(Infile1,'INT PC',k,'_',j,'_',t);
       end;
     end;
   end;

   Close(Infile1);
   Close(Outfile);
  End. {Main Program}
```

APPENDIX III

Listing of the Lindo Input Files

MODEL II

```
MIN Z
SUBJECT TO
!
Z1+Z2+Z3+Z4+Z5-Z=0
!
! 2 Cell Formation, 10 Different Part Types, 6 Different Machine Types
! Subcontracting Allowed
!
! CAPITAL INVESTMENT COST
!
16000Y1_1 + 20000Y2_1 + 18000Y3_1 + 24000Y4_1 + 14000Y5_1 + 17000Y6_1 +
16000Y1_2 + 20000Y2_2 + 18000Y3_2 + 24000Y4_2 + 14000Y5_2 + 17000Y6_2 - Z1=0
!
! INTERCELLULAR MOVEMENT COST
!
2000M1_1_1 + 2000M1_2_1 + 2000M2_2_1 + 2000M2_4_1 + 2000M3_3_1 +
2000M4_2_1 + 2000M4_4_1 + 2000M5_1_1 + 2000M5_2_1 + 2000M6_2_1 +
2000M6_3_1 + 2000M7_3_1 + 2000M8_1_1 + 2000M8_2_1 + 2000M8_4_1 +
2000M9_3_1 + 2000M9_5_1 + 2000M10_1_1 + 2000M10_2_1 + 2000M10_3_1 +
2000M1_1_2 + 2000M1_2_2 + 2000M2_2_2 + 2000M2_4_2 + 2000M3_3_2 +
2000M4_2_2 + 2000M4_4_2 + 2000M5_1_2 + 2000M5_2_2 + 2000M6_2_2 +
2000M6_3_2 + 2000M7_3_2 + 2000M8_1_2 + 2000M8_2_2 + 2000M8_4_2 +
2000M9_3_2 + 2000M9_5_2 + 2000M10_1_2 + 2000M10_2_2 + 2000M10_3_2 - Z2=0
!
! IDLE TIME COST
!
4 ID1_1 + 6 ID2_1 + 5 ID3_1 + 6 ID4_1 + 3 ID5_1 + 4 ID6_1 +
4 ID1_2 + 6 ID2_2 + 5 ID3_2 + 6 ID4_2 + 3 ID5_2 + 4 ID6_2 - Z3=0
!
! MACHINING COST
!
3600 Z1_1_1 + 8800 Z1_2_1 + 6600 Z1_4_1 + 14080 Z2_2_1 + 7920 Z2_4_1 +
5440 Z2_6_1 + 13200 Z3_3_1 + 14700 Z3_5_1 + 6600 Z4_2_1 + 7590 Z4_4_1 +
7200 Z4_6_1 + 9000 Z5_1_1 + 9537 Z5_2_1 + 7472 Z5_6_1 + 4774 Z6_2_1 +
4900 Z6_3_1 + 3920 Z6_5_1 + 7500 Z7_3_1 + 5670 Z7_5_1 + 6246 Z8_1_1 +
8514 Z8_2_1 + 7040 Z8_4_1 + 4696 Z8_6_1 + 6670 Z9_3_1 + 5138 Z9_5_1 +
5336 Z9_6_1 + 9000 Z10_1_1 + 11737 Z10_2_1 + 9340 Z10_3_1 + 7337 Z10_4_1 +
3600 Z1_1_2 + 8800 Z1_2_2 + 6600 Z1_4_2 + 14080 Z2_2_2 + 7920 Z2_4_2 +
5440 Z2_6_2 + 13200 Z3_3_2 + 14700 Z3_5_2 + 6600 Z4_2_2 + 7590 Z4_4_2 +
7200 Z4_6_2 + 9000 Z5_1_2 + 9537 Z5_2_2 + 7472 Z5_6_2 + 4774 Z6_2_2 +
4900 Z6_3_2 + 3920 Z6_5_2 + 7500 Z7_3_2 + 5670 Z7_5_2 + 6246 Z8_1_2 +
8514 Z8_2_2 + 7040 Z8_4_2 + 4696 Z8_6_2 + 6670 Z9_3_2 + 5138 Z9_5_2 +
5336 Z9_6_2 + 9000 Z10_1_2 + 11737 Z10_2_2 + 9340 Z10_3_2 + 7337 Z10_4_2 -Z4=0
!
! SUBCONTRACTING COST
!
60000S1 + 117600S2 + 151200S3 + 8000S4 + 100000S5 +
81200S6 + 84600S7 + 86400S8 + 94000S9 + 116200S10 - Z5 = 0
!
! Constraint Set 3.7
!
Z1_1_1 + Z1_1_2 <= 1
Z1_2_1 + Z1_2_2 <= 1
Z1_4_1 + Z1_4_2 <= 1
Z2_2_1 + Z2_2_2 <= 1
Z2_4_1 + Z2_4_2 <= 1
Z2_6_1 + Z2_6_2 <= 1
Z3_3_1 + Z3_3_2 <= 1
Z3_5_1 + Z3_5_2 <= 1
Z4_2_1 + Z4_2_2 <= 1
Z4_4_1 + Z4_4_2 <= 1
Z4_6_1 + Z4_6_2 <= 1
Z5_1_1 + Z5_1_2 <= 1
Z5_2_1 + Z5_2_2 <= 1
Z5_6_1 + Z5_6_2 <= 1
Z6_2_1 + Z6_2_2 <= 1
Z6_3_1 + Z6_3_2 <= 1
Z6_5_1 + Z6_5_2 <= 1
Z7_3_1 + Z7_3_2 <= 1
Z7_5_1 + Z7_5_2 <= 1
Z8_1_1 + Z8_1_2 <= 1
```

```
Z8_2_1 + Z8_2_2 <= 1
Z8_4_1 + Z8_4_2 <= 1
Z8_6_1 + Z8_6_2 <= 1
Z9_3_1 + Z9_3_2 <= 1
Z9_5_1 + Z9_5_2 <= 1
Z9_6_1 + Z9_6_2 <= 1
Z10_1_1 + Z10_1_2 <= 1
Z10_2_1 + Z10_2_2 <= 1
Z10_3_1 + Z10_3_2 <= 1
Z10_4_1 + Z10_4_2 <= 1
!
! Constraint Set 3.8
!
Z1_1_1 - Z1_2_1 - M1_1_1 <= 0
Z1_2_1 - Z1_4_1 - M1_2_1 <= 0
Z2_2_1 - Z2_4_1 - M2_2_1 <= 0
Z2_4_1 - Z2_6_1 - M2_4_1 <= 0
Z3_3_1 - Z3_5_1 - M3_3_1 <= 0
Z4_2_1 - Z4_4_1 - M4_2_1 <= 0
Z4_4_1 - Z4_6_1 - M4_4_1 <= 0
Z5_1_1 - Z5_2_1 - M5_1_1 <= 0
Z5_2_1 - Z5_6_1 - M5_2_1 <= 0
Z6_2_1 - Z6_3_1 - M6_2_1 <= 0
Z6_3_1 - Z6_5_1 - M6_3_1 <= 0
Z7_3_1 - Z7_5_1 - M7_3_1 <= 0
Z8_1_1 - Z8_2_1 - M8_1_1 <= 0
Z8_2_1 - Z8_4_1 - M8_2_1 <= 0
Z8_4_1 - Z8_6_1 - M8_4_1 <= 0
Z9_3_1 - Z9_5_1 - M9_3_1 <= 0
Z9_5_1 - Z9_6_1 - M9_5_1 <= 0
Z10_1_1 - Z10_2_1 - M10_1_1 <= 0
Z10_2_1 - Z10_3_1 - M10_2_1 <= 0
Z10_3_1 - Z10_4_1 - M10_3_1 <= 0
Z1_1_2 - Z1_2_2 - M1_1_2 <= 0
Z1_2_2 - Z1_4_2 - M1_2_2 <= 0
Z2_2_2 - Z2_4_2 - M2_2_2 <= 0
Z2_4_2 - Z2_6_2 - M2_4_2 <= 0
Z3_3_2 - Z3_5_2 - M3_3_2 <= 0
Z4_2_2 - Z4_4_2 - M4_2_2 <= 0
Z4_4_2 - Z4_6_2 - M4_4_2 <= 0
Z5_1_2 - Z5_2_2 - M5_1_2 <= 0
Z5_2_2 - Z5_6_2 - M5_2_2 <= 0
Z6_2_2 - Z6_3_2 - M6_2_2 <= 0
Z6_3_2 - Z6_5_2 - M6_3_2 <= 0
Z7_3_2 - Z7_5_2 - M7_3_2 <= 0
Z8_1_2 - Z8_2_2 - M8_1_2 <= 0
Z8_2_2 - Z8_4_2 - M8_2_2 <= 0
Z8_4_2 - Z8_6_2 - M8_4_2 <= 0
Z9_3_2 - Z9_5_2 - M9_3_2 <= 0
Z9_5_2 - Z9_6_2 - M9_5_2 <= 0
Z10_1_2 - Z10_2_2 - M10_1_2 <= 0
Z10_2_2 - Z10_3_2 - M10_2_2 <= 0
Z10_3_2 - Z10_4_2 - M10_3_2 <= 0
!
! Constraint Set 3.9
!
400Z1_1_1 + 1000Z5_1_1 + 694Z8_1_1 + 1000Z10_1_1 + ID1_1 - 2000Y1_1 = 0
800Z1_2_1 + 1280Z2_2_1 + 600Z4_2_1 + 867Z5_2_1 + 434Z6_2_1 +
774Z8_2_1 + 1067Z10_2_1 + ID2_1 - 2000Y2_1 = 0
1320Z3_3_1 + 490Z6_3_1 + 750Z7_3_1 + 667Z9_3_1 + 934Z10_3_1 +
ID3_1 - 2000Y3_1 = 0
600Z1_4_1 + 720Z2_4_1 + 690Z4_4_1 + 640Z8_4_1 +
667Z10_4_1 + ID4_1 - 2000Y4_1 = 0
2100Z3_5_1 + 560Z6_5_1 + 810Z7_5_1 + 734Z9_5_1 + ID5_1 - 2000Y5_1 = 0
680Z2_6_1 + 900Z4_6_1 + 934Z5_6_1 + 587Z8_6_1 +
667Z9_6_1 + ID6_1 - 2000Y6_1 = 0
400Z1_1_2 + 1000Z5_1_2 + 694Z8_1_2 + 1000Z10_1_2 + ID1_2 - 2000Y1_2 = 0
800Z1_2_2 + 1280Z2_2_2 + 600Z4_2_2 + 867Z5_2_2 +
434Z6_2_2 + 774Z8_2_2 + 1067Z10_2_2 + ID2_2 - 2000Y2_2 = 0
1320Z3_3_2 + 490Z6_3_2 + 750Z7_3_2 + 667Z9_3_2 +
934Z10_3_2 + ID3_2 - 2000Y3_2 = 0
600Z1_4_2 + 720Z2_4_2 + 690Z4_4_2 + 640Z8_4_2 +
667Z10_4_2 + ID4_2 - 2000Y4_2 = 0
2100Z3_5_2 + 560Z6_5_2 + 810Z7_5_2 + 734Z9_5_2 + ID5_2 - 2000Y5_2 = 0
```

```
660Z3_6_2 + 900Z4_6_2 + 934Z5_6_2 + 587Z8_6_2 +
667Z9_6_2 + 1D6_2 - 2000Y6_2 = 0
!
! Constraint Set 3.10
!
Z1_1_1 + Z1_2_1 + Z1_4_1 + Z1_1_2 + Z1_2_2 + Z1_4_2 + 3S1 = 3
Z2_2_1 + Z2_4_1 + Z2_6_1 + Z2_2_2 + Z2_4_2 + Z2_6_2 + 3S2 = 3
Z3_3_1 + Z3_5_1 + Z3_3_2 + Z3_5_2 + 2S3 = 2
Z4_2_1 + Z4_4_1 + Z4_6_1 + Z4_2_2 + Z4_4_2 + Z4_6_2 + 3S4 = 3
Z5_1_1 + Z5_2_1 + Z5_6_1 + Z5_1_2 + Z5_2_2 + Z5_6_2 + 3S5 = 3
Z6_2_1 + Z6_3_1 + Z6_5_1 + Z6_2_2 + Z6_3_2 + Z6_5_2 + 3S6 = 3
Z7_3_1 + Z7_5_1 + Z7_3_2 + Z7_5_2 + 2S7 = 2
Z8_1_1 + Z8_2_1 + Z8_4_1 + Z8_6_1 + Z8_1_2 + Z8_2_2 + Z8_4_2 +
Z8_6_2 + 4S8 = 4
Z9_3_1 + Z9_5_1 + Z9_6_1 + Z9_3_2 + Z9_5_2 + Z9_6_2 + 3S9 = 3
Z10_1_1 + Z10_2_1 + Z10_3_1 + Z10_4_1 + Z10_1_2 + Z10_2_2 +
Z10_3_2 + Z10_4_2 + 4S10 = 4
!
! Constraint Set 3.11
!
Y1_1 + Y2_1 + Y3_1 + Y4_1 + Y5_1 + Y6_1 >= 6
Y1_2 + Y2_2 + Y3_2 + Y4_2 + Y5_2 + Y6_2 >= 7
Y1_1 + Y2_1 + Y3_1 + Y4_1 + Y5_1 + Y6_1 <= 7
Y1_2 + Y2_2 + Y3_2 + Y4_2 + Y5_2 + Y6_2 <= 8

END
```

(Constraint Set 3.12)
INTEGER S_p $Z_{i,k,p}$ $M_{i,k,j}$

(Constraint Set 3.13)
GIN Y_{ij}

MODEL IV

```
MIN Z
SUBJECT TO
Z1+Z2+Z3+Z4+Z5+Z6-Z=0
! 2 Cell Formation, 10 Part Types, 7 Machine Types, 3 Planning Periods
!
! CAPITAL INVESTMENT COST
18000Y1_1_1 + 20000Y2_1_1 + 22000Y3_1_1 + 24000Y4_1_1 + 23000Y5_1_1 + 19000Y6_1_1 +
20000Y7_1_1 + 18000Y1_2_1 + 20000Y2_2_1 + 22000Y3_2_1 + 24000Y4_2_1 + 23000Y5_2_1 +
19000Y6_2_1 + 20000Y7_2_1 + 18000Y1_1_2 + 20000Y2_1_2 + 22000Y3_1_2 + 24000Y4_1_2 +
23000Y5_1_2 + 19000Y6_1_2 + 20000Y7_1_2 + 18000Y1_2_2 + 20000Y2_2_2 + 22000Y3_2_2 +
24000Y4_2_2 + 23000Y5_2_2 + 19000Y6_2_2 + 20000Y7_2_2 + 18000Y1_1_3 + 20000Y2_1_3 +
22000Y3_1_3 + 24000Y4_1_3 + 23000Y5_1_3 + 19000Y6_1_3 + 20000Y7_1_3 + 18000Y1_2_3 +
20000Y2_2_3 + 22000Y3_2_3 + 24000Y4_2_3 + 23000Y5_2_3 + 19000Y6_2_3 + 20000Y7_2_3 - Z1 = 0
!
! INTERCELLULAR MOVEMENT COST
2000M1_1_11 + 2000M1_3_11 + 2000M2_2_11 + 2000M3_1_11 + 2000M4_2_11 + 2000M5_1_11 +
2000M6_4_11 + 2000M6_6_11 + 2000M7_2_11 + 2000M7_5_11 + 2000M8_4_11 + 2000M9_2_11 +
2000M9_5_11 + 2000M10_4_11 + 2000M1_1_21 + 2000M1_3_21 + 2000M2_2_21 + 2000M3_1_21 +
2000M4_2_21 + 2000M5_1_21 + 2000M6_4_21 + 2000M6_6_21 + 2000M7_2_21 + 2000M7_5_21 +
2000M8_4_21 + 2000M9_2_21 + 2000M9_5_21 + 2000M10_4_21 + 2000M1_1_12 + 2000M1_3_12 +
2000M2_2_12 + 2000M3_1_12 + 2000M4_2_12 + 2000M5_1_12 + 2000M6_4_12 + 2000M6_6_12 +
2000M7_2_12 + 2000M7_5_12 + 2000M8_4_12 + 2000M9_2_12 + 2000M9_5_12 + 2000M10_4_12 +
2000M1_1_22 + 2000M1_3_22 + 2000M2_2_22 + 2000M3_1_22 + 2000M4_2_22 + 2000M5_1_22 +
2000M6_4_22 + 2000M6_6_22 + 2000M7_2_22 + 2000M7_5_22 + 2000M8_4_22 + 2000M9_2_22 +
2000M9_5_22 + 2000M10_4_22 + 2000M1_1_13 + 2000M1_3_13 + 2000M2_2_13 + 2000M3_1_13 +
2000M4_2_13 + 2000M5_1_13 + 2000M6_4_13 + 2000M6_6_13 + 2000M7_2_13 + 2000M7_5_13 +
2000M8_4_13 + 2000M9_2_13 + 2000M9_5_13 + 2000M10_4_13 + 2000M1_1_23 + 2000M1_3_23 +
2000M2_2_23 + 2000M3_1_23 + 2000M4_2_23 + 2000M5_1_23 + 2000M6_4_23 + 2000M6_6_23 +
2000M7_2_23 + 2000M7_5_23 + 2000M8_4_23 + 2000M9_2_23 + 2000M9_5_23 + 2000M10_4_23 - Z2
= 0
!
! IDLE TIME COST
4ID1_11 + 6ID2_11 + 4ID3_11 + 3ID4_11 + 4ID5_11 + 3ID6_11 + 5ID7_11 + 4ID1_21 + 6ID2_21 +
4ID3_21 + 3ID4_21 + 4ID5_21 + 3ID6_21 + 5ID7_21 + 4ID1_12 + 6ID2_12 + 4ID3_12 + 3ID4_12 +
4ID5_12 + 3ID6_12 + 5ID7_12 + 4ID1_22 + 6ID2_22 + 4ID3_22 + 3ID4_22 + 4ID5_22 + 3ID6_22 +
5ID7_22 + 4ID1_13 + 6ID2_13 + 4ID3_13 + 3ID4_13 + 4ID5_13 + 3ID6_13 + 5ID7_13 + 4ID1_23 +
6ID2_23 + 4ID3_23 + 3ID4_23 + 4ID5_23 + 3ID6_23 + 5ID7_23 - Z3 = 0
!
!
! MACHINING COST
6300Z1_1_11 + 7315Z1_3_11 + 6160Z1_5_11 + 7200Z2_2_11 + 9192Z2_4_11 + 4590Z3_1_11 +
8640Z3_4_11 + 6600Z4_2_11 + 9672Z4_4_11 + 5985Z5_1_11 + 8360Z5_3_11 + 6456Z6_4_11 +
6960Z6_6_11 + 5409Z6_7_11 + 10800Z7_2_11 + 6600Z7_5_11 + 5000Z7_6_11 + 8796Z8_4_11 +
6230Z8_6_11 + 11496Z9_2_11 + 8008Z9_5_11 + 6210Z9_7_11 + 8400Z10_4_11 + 5994Z10_7_11 +
6300Z1_1_21 + 7315Z1_3_21 + 6160Z1_5_21 + 7200Z2_2_21 + 9192Z2_4_21 + 4590Z3_1_21 +
8640Z3_4_21 + 6600Z4_2_21 + 9672Z4_4_21 + 5985Z5_1_21 + 8360Z5_3_21 + 6456Z6_4_21 +
6960Z6_6_21 + 5409Z6_7_21 + 10800Z7_2_21 + 6600Z7_5_21 + 5000Z7_6_21 + 8796Z8_4_21 +
6230Z8_6_21 + 11496Z9_2_21 + 8008Z9_5_21 + 6210Z9_7_21 + 8400Z10_4_21 + 5994Z10_7_21 +
6597Z1_1_12+ 7656Z1_3_12 + 6446Z1_5_12 + 7560Z2_2_12 + 9660Z2_4_12 + 5094Z3_1_12 +
9600Z3_4_12 + 6900Z4_2_12 + 10116Z4_4_12 + 6300Z5_1_12 + 8800Z5_3_12 + 5436Z6_4_12 +
5860Z6_6_12 + 4554Z6_7_12 + 4860Z7_2_12 + 2970Z7_5_12 + 2250Z7_6_12 + 7596Z8_4_12 +
5380Z8_6_12 + 4500Z9_2_12 + 3135Z9_5_12 + 2430Z9_7_12 + 7560Z10_4_12 + 5400Z10_7_12 +
6597Z1_1_22 + 7656Z1_3_22 + 6446Z1_5_22 + 7560Z2_2_22 + 9660Z2_4_22 + 5094Z3_1_22 +
9600Z3_4_22 + 6900Z4_2_22 + 10116Z4_4_22 + 6300Z5_1_22 + 8800Z5_3_22 + 5436Z6_4_22 +
5860Z6_6_22 + 4554Z6_7_22 + 4860Z7_2_22 + 2970Z7_5_22 + 2250Z7_6_22 + 7596Z8_4_22 +
5380Z8_6_22 + 4500Z9_2_22 + 3135Z9_5_22 + 2430Z9_7_22 + 7560Z10_4_22 + 5400Z10_7_22 +
6300Z1_1_13 + 7315Z1_3_13 + 6160Z1_5_13 + 6480Z2_2_13 + 8280Z2_4_13 + 3825Z3_1_13 +
7200Z3_4_13 + 5700Z4_2_13 + 8352Z4_4_13 + 6615Z5_1_13 + 9240Z5_3_13 + 4416Z6_4_13 +
4760Z6_6_13 + 3699Z6_7_13 + 3780Z7_2_13 + 2310Z7_5_13 + 1750Z7_6_13 + 6000Z8_4_13 +
4250Z8_6_13 + 3000Z9_2_13 + 2090Z9_5_13 + 1620Z9_7_13 + 5880Z10_4_13 + 4194Z10_7_13 +
6300Z1_1_23 + 7315Z1_3_23 + 6160Z1_5_23 + 6480Z2_2_23 + 8280Z2_4_23 + 3825Z3_1_23 +
7200Z3_4_23 + 5700Z4_2_23 + 8352Z4_4_23 + 6615Z5_1_23 + 9240Z5_3_23 + 4416Z6_4_23 +
4760Z6_6_23 + 3699Z6_7_23 + 3780Z7_2_23 + 2310Z7_5_23 + 1750Z7_6_23 + 6000Z8_4_23 +
4250Z8_6_23 + 3000Z9_2_23 + 2090Z9_5_23 + 1620Z9_7_23 + 5880Z10_4_23 +
4194Z10_7_23 - Z4 = 0
! SUBCONTRACTING COST
!
9450S1_1 + 7000S2_1 + 7200S3_1 + 7260S4_1 + 6840S5_1 + 7600S6_1 + 8400S7_1 +
9680S8_1 + 10350S9_1 + 8000S10_1 + 9900S1_2 + 7350S2_2 + 8000S3_2 + 7590S4_2 +
7200S5_2 + 6400S6_2 + 3780S7_2 + 8360S8_2 + 4050S9_2 + 7200S10_2 + 9450S1_3 +
6300S2_3 + 6000S3_3 + 6270S4_3 + 7560S5_3 + 5200S6_3 + 2940S7_3 + 6600S8_3 +
2700S9_3 + 5600S10_3 - Z5 = 0
! RELOCATION COST
!
5000PC1_1_1 + 5000NC1_1_1 + 5000PC2_1_1 + 5000NC2_1_1 + 5000PC3_1_1 + 5000NC3_1_1 +
5000PC4_1_1 + 5000NC4_1_1 + 5000PC5_1_1 + 5000NC5_1_1 + 5000PC6_1_1 + 5000NC6_1_1 +
5000PC7_1_1 + 5000NC7_1_1 + 5000PC1_2_1 + 5000NC1_2_1 + 5000PC2_2_1 + 5000NC2_2_1 +
5000PC3_2_1 + 5000NC3_2_1 + 5000PC4_2_1 + 5000NC4_2_1 + 5000PC5_2_1 + 5000NC5_2_1 +
5000PC6_2_1 + 5000NC6_2_1 + 5000PC7_2_1 + 5000NC7_2_1 + 5000PC1_1_2 + 5000NC1_1_2 +
```

```
5000PC2_1_2 + 5000NC2_1_2 + 5000PC3_1_2 + 5000NC3_1_2 + 5000PC4_1_2 + 5000NC4_1_2 +
5000PC5_1_2 + 5000NC5_1_2 + 5000PC6_1_2 + 5000NC6_1_2 + 5000PC7_1_2 + 5000NC7_1_2 +
5000PC1_2_2 + 5000NC1_2_2 + 5000PC2_2_2 + 5000NC2_2_2 + 5000PC3_2_2 + 5000NC3_2_2 +
5000PC4_2_2 + 5000NC4_2_2 + 5000PC5_2_2 + 5000NC5_2_2 + 5000PC6_2_2 + 5000NC6_2_2 +
5000PC7_2_2 + 5000NC7_2_2 - Z6 = 0
!
! Constraint Set (3.21)
Z1_1_11 + Z1_1_21 <= 1
Z1_3_11 + Z1_3_21 <= 1
Z1_5_11 + Z1_5_21 <= 1
Z2_2_11 + Z2_2_21 <= 1
Z2_4_11 + Z2_4_21 <= 1
Z3_1_11 + Z3_1_21 <= 1
Z3_4_11 + Z3_4_21 <= 1
Z4_2_11 + Z4_2_21 <= 1
Z4_4_11 + Z4_4_21 <= 1
Z5_1_11 + Z5_1_21 <= 1
Z5_3_11 + Z5_3_21 <= 1
Z6_4_11 + Z6_4_21 <= 1
Z6_6_11 + Z6_6_21 <= 1
Z6_7_11 + Z6_7_21 <= 1
Z7_2_11 + Z7_2_21 <= 1
Z7_5_11 + Z7_5_21 <= 1
Z7_6_11 + Z7_6_21 <= 1
Z8_4_11 + Z8_4_21 <= 1
Z8_6_11 + Z8_6_21 <= 1
Z9_2_11 + Z9_2_21 <= 1
Z9_5_11 + Z9_5_21 <= 1
Z9_7_11 + Z9_7_21 <= 1
Z10_4_11 + Z10_4_21 <= 1
Z10_7_11 + Z10_7_21 <= 1
Z1_1_12 + Z1_1_22 <= 1
Z1_3_12 + Z1_3_22 <= 1
Z1_5_12 + Z1_5_22 <= 1
Z2_2_12 + Z2_2_22 <= 1
Z2_4_12 + Z2_4_22 <= 1
Z3_1_12 + Z3_1_22 <= 1
Z3_4_12 + Z3_4_22 <= 1
Z4_2_12 + Z4_2_22 <= 1
Z4_4_12 + Z4_4_22 <= 1
Z5_1_12 + Z5_1_22 <= 1
Z5_3_12 + Z5_3_22 <= 1
Z6_4_12 + Z6_4_22 <= 1
Z6_6_12 + Z6_6_22 <= 1
Z6_7_12 + Z6_7_22 <= 1
Z7_2_12 + Z7_2_22 <= 1
Z7_5_12 + Z7_5_22 <= 1
Z7_6_12 + Z7_6_22 <= 1
Z8_4_12 + Z8_4_22 <= 1
Z8_6_12 + Z8_6_22 <= 1
Z9_2_12 + Z9_2_22 <= 1
Z9_5_12 + Z9_5_22 <= 1
Z9_7_12 + Z9_7_22 <= 1
Z10_4_12 + Z10_4_22 <= 1
Z10_7_12 + Z10_7_22 <= 1
Z1_1_13 + Z1_1_23 <= 1
Z1_3_13 + Z1_3_23 <= 1
Z1_5_13 + Z1_5_23 <= 1
Z2_2_13 + Z2_2_23 <= 1
Z2_4_13 + Z2_4_23 <= 1
Z3_1_13 + Z3_1_23 <= 1
Z3_4_13 + Z3_4_23 <= 1
Z4_2_13 + Z4_2_23 <= 1
Z4_4_13 + Z4_4_23 <= 1
Z5_1_13 + Z5_1_23 <= 1
Z5_3_13 + Z5_3_23 <= 1
Z6_4_13 + Z6_4_23 <= 1
Z6_6_13 + Z6_6_23 <= 1
Z6_7_13 + Z6_7_23 <= 1
Z7_2_13 + Z7_2_23 <= 1
Z7_5_13 + Z7_5_23 <= 1
Z7_6_13 + Z7_6_23 <= 1
Z8_4_13 + Z8_4_23 <= 1
Z8_6_13 + Z8_6_23 <= 1
Z9_2_13 + Z9_2_23 <= 1
Z9_5_13 + Z9_5_23 <= 1
Z9_7_13 + Z9_7_23 <= 1
Z10_4_13 + Z10_4_23 <= 1
Z10_7_13 + Z10_7_23 <= 1
!
! Constraint Set (3.22)
Z1_1_11 - Z1_3_11 - M1_1_11 <= 0
```

Z1_3_11 - Z1_5_11 - M1_3_11 <= 0
Z2_2_11 - Z2_4_11 - M2_2_11 <= 0
Z3_1_11 - Z3_4_11 - M3_1_11 <= 0
Z4_2_11 - Z4_4_11 - M4_2_11 <= 0
Z5_1_11 - Z5_3_11 - M5_1_11 <= 0
Z6_4_11 - Z6_6_11 - M6_4_11 <= 0
Z6_6_11 - Z6_7_11 - M6_6_11 <= 0
Z7_2_11 - Z7_5_11 - M7_2_11 <= 0
Z7_5_11 - Z7_6_11 - M7_5_11 <= 0
Z8_4_11 - Z8_6_11 - M8_4_11 <= 0
Z9_2_11 - Z9_5_11 - M9_2_11 <= 0
Z9_5_11 - Z9_7_11 - M9_5_11 <= 0
Z10_4_11 - Z10_7_11 - M10_4_11 <= 0
Z1_1_21 - Z1_3_21 - M1_1_21 <= 0
Z1_3_21 - Z1_5_21 - M1_3_21 <= 0
Z2_2_21 - Z2_4_21 - M2_2_21 <= 0
Z3_1_21 - Z3_4_21 - M3_1_21 <= 0
Z4_2_21 - Z4_4_21 - M4_2_21 <= 0
Z5_1_21 - Z5_3_21 - M5_1_21 <= 0
Z6_4_21 - Z6_6_21 - M6_4_21 <= 0
Z6_6_21 - Z6_7_21 - M6_6_21 <= 0
Z7_2_21 - Z7_5_21 - M7_2_21 <= 0
Z7_5_21 - Z7_6_21 - M7_5_21 <= 0
Z8_4_21 - Z8_6_21 - M8_4_21 <= 0
Z9_2_21 - Z9_5_21 - M9_2_21 <= 0
Z9_5_21 - Z9_7_21 - M9_5_21 <= 0
Z10_4_21 - Z10_7_21 - M10_4_21 <= 0
Z1_1_12 - Z1_3_12 - M1_1_12 <= 0
Z1_3_12 - Z1_5_12 - M1_3_12 <= 0
Z2_2_12 - Z2_4_12 - M2_2_12 <= 0
Z3_1_12 - Z3_4_12 - M3_1_12 <= 0
Z4_2_12 - Z4_4_12 - M4_2_12 <= 0
Z5_1_12 - Z5_3_12 - M5_1_12 <= 0
Z6_4_12 - Z6_6_12 - M6_4_12 <= 0
Z6_6_12 - Z6_7_12 - M6_6_12 <= 0
Z7_2_12 - Z7_5_12 - M7_2_12 <= 0
Z7_5_12 - Z7_6_12 - M7_5_12 <= 0
Z8_4_12 - Z8_6_12 - M8_4_12 <= 0
Z9_2_12 - Z9_5_12 - M9_2_12 <= 0
Z9_5_12 - Z9_7_12 - M9_5_12 <= 0
Z10_4_12 - Z10_7_12 - M10_4_12 <= 0
Z1_1_22 - Z1_3_22 - M1_1_22 <= 0
Z1_3_22 - Z1_5_22 - M1_3_22 <= 0
Z2_2_22 - Z2_4_22 - M2_2_22 <= 0
Z3_1_22 - Z3_4_22 - M3_1_22 <= 0
Z4_2_22 - Z4_4_22 - M4_2_22 <= 0
Z5_1_22 - Z5_3_22 - M5_1_22 <= 0
Z6_4_22 - Z6_6_22 - M6_4_22 <= 0
Z6_6_22 - Z6_7_22 - M6_6_22 <= 0
Z7_2_22 - Z7_5_22 - M7_2_22 <= 0
Z7_5_22 - Z7_6_22 - M7_5_22 <= 0
Z8_4_22 - Z8_6_22 - M8_4_22 <= 0
Z9_2_22 - Z9_5_22 - M9_2_22 <= 0
Z9_5_22 - Z9_7_22 - M9_5_22 <= 0
Z10_4_22 - Z10_7_22 - M10_4_22 <= 0
Z1_1_13 - Z1_3_13 - M1_1_13 <= 0
Z1_3_13 - Z1_5_13 - M1_3_13 <= 0
Z2_2_13 - Z2_4_13 - M2_2_13 <= 0
Z3_1_13 - Z3_4_13 - M3_1_13 <= 0
Z4_2_13 - Z4_4_13 - M4_2_13 <= 0
Z5_1_13 - Z5_3_13 - M5_1_13 <= 0
Z6_4_13 - Z6_6_13 - M6_4_13 <= 0
Z6_6_13 - Z6_7_13 - M6_6_13 <= 0
Z7_2_13 - Z7_5_13 - M7_2_13 <= 0
Z7_5_13 - Z7_6_13 - M7_5_13 <= 0
Z8_4_13 - Z8_6_13 - M8_4_13 <= 0
Z9_2_13 - Z9_5_13 - M9_2_13 <= 0
Z9_5_13 - Z9_7_13 - M9_5_13 <= 0
Z10_4_13 - Z10_7_13 - M10_4_13 <= 0
Z1_1_23 - Z1_3_23 - M1_1_23 <= 0
Z1_3_23 - Z1_5_23 - M1_3_23 <= 0
Z2_2_23 - Z2_4_23 - M2_2_23 <= 0
Z3_1_23 - Z3_4_23 - M3_1_23 <= 0
Z4_2_23 - Z4_4_23 - M4_2_23 <= 0
Z5_1_23 - Z5_3_23 - M5_1_23 <= 0
Z6_4_23 - Z6_6_23 - M6_4_23 <= 0
Z6_6_23 - Z6_7_23 - M6_6_23 <= 0
Z7_2_23 - Z7_5_23 - M7_2_23 <= 0
Z7_5_23 - Z7_6_23 - M7_5_23 <= 0
Z8_4_23 - Z8_6_23 - M8_4_23 <= 0
Z9_2_23 - Z9_5_23 - M9_2_23 <= 0
Z9_5_23 - Z9_7_23 - M9_5_23 <= 0

```
ZI0_4_23 - ZI0_7_23 - MI0_4_23 <= 0
!
! Constraint Set (3.23)
700ZI_1_11 + 510Z3_1_11 + 665Z5_1_11 + ID1_11 - 2000Y1_1_1 = 0
600Z2_2_11 + 550Z4_2_11 + 900Z7_2_11 + 958Z9_2_11 + ID2_11 - 2000Y2_1_1 = 0
665ZI_3_11 + 760Z5_3_11 + ID3_11 - 2000Y3_1_1 = 0
766Z2_4_11 + 720Z3_4_11 + 806Z4_4_11 + 538Z6_4_11 + 733Z8_4_11 + 700ZI0_4_11 +
ID4_11 - 2000Y4_1_1 = 0
560ZI_5_11 + 600Z7_5_11 + 728Z9_5_11 + ID5_11 - 2000Y5_1_1 = 0
696Z6_6_11 + 500Z7_6_11 + 623Z8_6_11 + ID6_11 - 2000Y6_1_1 = 0
601Z6_7_11 + 690Z9_7_11 + 666ZI0_7_11 + ID7_11 - 2000Y7_1_1 = 0
700ZI_1_21 + 510Z3_1_21 + 665Z5_1_21 + ID1_21 - 2000Y1_2_1 = 0
600Z2_2_21 + 550Z4_2_21 + 900Z7_2_21 + 958Z9_2_21 + ID2_21 - 2000Y2_2_1 = 0
665ZI_3_21 + 760Z5_3_21 + ID3_21 - 2000Y3_2_1 = 0
766Z2_4_21 + 720Z3_4_21 + 806Z4_4_21 + 538Z6_4_21 + 733Z8_4_21 + 700ZI0_4_21 +
ID4_21 - 2000Y4_2_1 = 0
560ZI_5_21 + 600Z7_5_21 + 728Z9_5_21 + ID5_21 - 2000Y5_2_1 = 0
696Z6_6_21 + 500Z7_6_21 + 623Z8_6_21 + ID6_21 - 2000Y6_2_1 = 0
601Z6_7_21 + 690Z9_7_21 + 666ZI0_7_21 + ID7_21 - 2000Y7_2_1 = 0
733ZI_1_12 + 566Z3_1_12 + 700Z5_1_12 + ID1_12 - 2000Y1_1_2 = 0
630Z2_2_12 + 575Z4_2_12 + 405Z7_2_12 + 375Z9_2_12 + ID2_12 - 2000Y2_1_2 = 0
696ZI_3_12 + 800Z5_3_12 + ID3_12 - 2000Y3_1_2 = 0
805Z2_4_12 + 800Z3_4_12 + 843Z4_4_12 + 453Z6_4_12 + 633Z8_4_12 + 630ZI0_4_12 +
ID4_12 - 2000Y4_1_2 = 0
586ZI_5_12 + 270Z7_5_12 + 285Z9_5_12 + ID5_12 - 2000Y5_1_2 = 0
586Z6_6_12 + 225Z7_6_12 + 538Z8_6_12 + ID6_12 - 2000Y6_1_2 = 0
506Z6_7_12 + 270Z9_7_12 + 600ZI0_7_12 + ID7_12 - 2000Y7_1_2 = 0
733ZI_1_22 + 566Z3_1_22 + 700Z5_1_22 + ID1_22 - 2000Y1_2_2 = 0
630Z2_2_22 + 575Z4_2_22 + 405Z7_2_22 + 375Z9_2_22 + ID2_22 - 2000Y2_2_2 = 0
696ZI_3_22 + 800Z5_3_22 + ID3_22 - 2000Y3_2_2 = 0
805Z2_4_22 + 800Z3_4_22 + 843Z4_4_22 + 453Z6_4_22 + 633Z8_4_22 + 630ZI0_4_22 +
ID4_22 - 2000Y4_2_2 = 0
586ZI_5_22 + 270Z7_5_22 + 285Z9_5_22 + ID5_22 - 2000Y5_2_2 = 0
586Z6_6_22 + 225Z7_6_22 + 538Z8_6_22 + ID6_22 - 2000Y6_2_2 = 0
506Z6_7_22 + 270Z9_7_22 + 600ZI0_7_22 + ID7_22 - 2000Y7_2_2 = 0
700ZI_1_13 + 425Z3_1_13 + 735Z5_1_13 + ID1_13 - 2000Y1_1_3 = 0
540Z2_2_13 + 475Z4_2_13 + 315Z7_2_13 + 250Z9_2_13 + ID2_13 - 2000Y2_1_3 = 0
665ZI_3_13 + 840Z5_3_13 + ID3_13 - 2000Y3_1_3 = 0
690Z2_4_13 + 600Z3_4_13 + 696Z4_4_13 + 368Z6_4_13 + 500Z8_4_13 + 490ZI0_4_13 +
ID4_13 - 2000Y4_1_3 = 0
560ZI_5_13 + 210Z7_5_13 + 190Z9_5_13 + ID5_13 - 2000Y5_1_3 = 0
476Z6_6_13 + 175Z7_6_13 + 425Z8_6_13 + ID6_13 - 2000Y6_1_3 = 0
411Z6_7_13 + 180Z9_7_13 + 466ZI0_7_13 + ID7_13 - 2000Y7_1_3 = 0
700ZI_1_23 + 425Z3_1_23 + 735Z5_1_23 + ID1_23 - 2000Y1_2_3 = 0
540Z2_2_23 + 475Z4_2_23 + 315Z7_2_23 + 250Z9_2_23 + ID2_23 - 2000Y2_2_3 = 0
665ZI_3_23 + 840Z5_3_23 + ID3_23 - 2000Y3_2_3 = 0
690Z2_4_23 + 600Z3_4_23 + 696Z4_4_23 + 368Z6_4_23 + 500Z8_4_23 + 490ZI0_4_23 +
ID4_23 - 2000Y4_2_3 = 0
560ZI_5_23 + 210Z7_5_23 + 190Z9_5_23 + ID5_23 - 2000Y5_2_3 = 0
476Z6_6_23 + 175Z7_6_23 + 425Z8_6_23 + ID6_23 - 2000Y6_2_3 = 0
411Z6_7_23 + 180Z9_7_23 + 466ZI0_7_23 + ID7_23 - 2000Y7_2_3 = 0
!
! Constraint Set (3.24)
YI_1_1 + Y2_1_1 + Y3_1_1 + Y4_1_1 + Y5_1_1 + Y6_1_1 + Y7_1_1 >=3
YI_2_1 + Y2_2_1 + Y3_2_1 + Y4_2_1 + Y5_2_1 + Y6_2_1 + Y7_2_1 >=3
YI_1_2 + Y2_1_2 + Y3_1_2 + Y4_1_2 + Y5_1_2 + Y6_1_2 + Y7_1_2 >=3
YI_2_2 + Y2_2_2 + Y3_2_2 + Y4_2_2 + Y5_2_2 + Y6_2_2 + Y7_2_2 >=3
YI_1_3 + Y2_1_3 + Y3_1_3 + Y4_1_3 + Y5_1_3 + Y6_1_3 + Y7_1_3 >=3
YI_2_3 + Y2_2_3 + Y3_2_3 + Y4_2_3 + Y5_2_3 + Y6_2_3 + Y7_2_3 >=3
YI_1_1 + Y2_1_1 + Y3_1_1 + Y4_1_1 + Y5_1_1 + Y6_1_1 + Y7_1_1 <=5
YI_2_1 + Y2_2_1 + Y3_2_1 + Y4_2_1 + Y5_2_1 + Y6_2_1 + Y7_2_1 <=5
YI_1_2 + Y2_1_2 + Y3_1_2 + Y4_1_2 + Y5_1_2 + Y6_1_2 + Y7_1_2 <=5
YI_2_2 + Y2_2_2 + Y3_2_2 + Y4_2_2 + Y5_2_2 + Y6_2_2 + Y7_2_2 <=5
YI_1_3 + Y2_1_3 + Y3_1_3 + Y4_1_3 + Y5_1_3 + Y6_1_3 + Y7_1_3 <=5
YI_2_3 + Y2_2_3 + Y3_2_3 + Y4_2_3 + Y5_2_3 + Y6_2_3 + Y7_2_3 <=5
!
! Constraint Set (3.25)
YI_1_2 - YI_1_1 - PCI_1_1 + NCI_1_1 = 0
Y2_1_2 - Y2_1_1 - PC2_1_1 + NC2_1_1 = 0
Y3_1_2 - Y3_1_1 - PC3_1_1 + NC3_1_1 = 0
Y4_1_2 - Y4_1_1 - PC4_1_1 + NC4_1_1 = 0
Y5_1_2 - Y5_1_1 - PC5_1_1 + NC5_1_1 = 0
Y6_1_2 - Y6_1_1 - PC6_1_1 + NC6_1_1 = 0
Y7_1_2 - Y7_1_1 - PC7_1_1 + NC7_1_1 = 0
YI_2_2 - YI_2_1 - PCI_2_1 + NCI_2_1 = 0
Y2_2_2 - Y2_2_1 - PC2_2_1 + NC2_2_1 = 0
Y3_2_2 - Y3_2_1 - PC3_2_1 + NC3_2_1 = 0
Y4_2_2 - Y4_2_1 - PC4_2_1 + NC4_2_1 = 0
Y5_2_2 - Y5_2_1 - PC5_2_1 + NC5_2_1 = 0
Y6_2_2 - Y6_2_1 - PC6_2_1 + NC6_2_1 = 0
Y7_2_2 - Y7_2_1 - PC7_2_1 + NC7_2_1 = 0
YI_1_3 - YI_1_2 - PCI_1_2 + NCI_1_2 = 0
```

```
Y2_1_3 - Y2_1_2 - PC2_1_2 + NC2_1_2 = 0
Y3_1_3 - Y3_1_2 - PC3_1_2 + NC3_1_2 = 0
Y4_1_3 - Y4_1_2 - PC4_1_2 + NC4_1_2 = 0
Y5_1_3 - Y5_1_2 - PC5_1_2 + NC5_1_2 = 0
Y6_1_3 - Y6_1_2 - PC6_1_2 + NC6_1_2 = 0
Y7_1_3 - Y7_1_2 - PC7_1_2 + NC7_1_2 = 0
Y1_2_3 - Y1_2_2 - PC1_2_2 + NC1_2_2 = 0
Y2_2_3 - Y2_2_2 - PC2_2_2 + NC2_2_2 = 0
Y3_2_3 - Y3_2_2 - PC3_2_2 + NC3_2_2 = 0
Y4_2_3 - Y4_2_2 - PC4_2_2 + NC4_2_2 = 0
Y5_2_3 - Y5_2_2 - PC5_2_2 + NC5_2_2 = 0
Y6_2_3 - Y6_2_2 - PC6_2_2 + NC6_2_2 = 0
Y7_2_3 - Y7_2_2 - PC7_2_2 + NC7_2_2 = 0
!
! Constraint Set (3.26)
Z1_1_11 + Z1_3_11 + Z1_5_11 + Z1_1_21 + Z1_3_21 + Z1_5_21 + 3S1_1 = 3
Z2_2_11 + Z2_4_11 + Z2_2_21 + Z2_4_21 + 2S2_1 = 2
Z3_1_11 + Z3_4_11 + Z3_1_21 + Z3_4_21 + 2S3_1 = 2
Z4_2_11 + Z4_4_11 + Z4_2_21 + Z4_4_21 + 2S4_1 = 2
Z5_1_11 + Z5_3_11 + Z5_1_21 + Z5_3_21 + 2S5_1 = 2
Z6_4_11 + Z6_6_11 + Z6_7_11 + Z6_4_21 + Z6_6_21 + Z6_7_21 + 3S6_1 = 3
Z7_2_11 + Z7_5_11 + Z7_6_11 + Z7_2_21 + Z7_5_21 + Z7_6_21 + 3S7_1 = 3
Z8_4_11 + Z8_6_11 + Z8_4_21 + Z8_6_21 + 2S8_1 = 2
Z9_2_11 + Z9_5_11 + Z9_7_11 + Z9_2_21 + Z9_5_21 + Z9_7_21 + 3S9_1 = 3
Z10_4_11 + Z10_7_11 + Z10_4_21 + Z10_7_21 + 2S10_1 = 2
Z1_1_12 + Z1_3_12 + Z1_5_12 + Z1_1_22 + Z1_3_22 + Z1_5_22 + 3S1_2 = 3
Z2_2_12 + Z2_4_12 + Z2_2_22 + Z2_4_22 + 2S2_2 = 2
Z3_1_12 + Z3_4_12 + Z3_1_22 + Z3_4_22 + 2S3_2 = 2
Z4_2_12 + Z4_4_12 + Z4_2_22 + Z4_4_22 + 2S4_2 = 2
Z5_1_12 + Z5_3_12 + Z5_1_22 + Z5_3_22 + 2S5_2 = 2
Z6_4_12 + Z6_6_12 + Z6_7_12 + Z6_4_22 + Z6_6_22 + Z6_7_22 + 3S6_2 = 3
Z7_2_12 + Z7_5_12 + Z7_6_12 + Z7_2_22 + Z7_5_22 + Z7_6_22 + 3S7_2 = 3
Z8_4_12 + Z8_6_12 + Z8_4_22 + Z8_6_22 + 2S8_2 = 2
Z9_2_12 + Z9_5_12 + Z9_7_12 + Z9_2_22 + Z9_5_22 + Z9_7_22 + 3S9_2 = 3
Z10_4_12 + Z10_7_12 + Z10_4_22 + Z10_7_22 + 2S10_2 = 2
Z1_1_13 + Z1_3_13 + Z1_5_13 + Z1_1_23 + Z1_3_23 + Z1_5_23 + 3S1_3 = 3
Z2_2_13 + Z2_4_13 + Z2_2_23 + Z2_4_23 + 2S2_3 = 2
Z3_1_13 + Z3_4_13 + Z3_1_23 + Z3_4_23 + 2S3_3 = 2
Z4_2_13 + Z4_4_13 + Z4_2_23 + Z4_4_23 + 2S4_3 = 2
Z5_1_13 + Z5_3_13 + Z5_1_23 + Z5_3_23 + 2S5_3 = 2
Z6_4_13 + Z6_6_13 + Z6_7_13 + Z6_4_23 + Z6_6_23 + Z6_7_23 + 3S6_3 = 3
Z7_2_13 + Z7_5_13 + Z7_6_13 + Z7_2_23 + Z7_5_23 + Z7_6_23 + 3S7_3 = 3
Z8_4_13 + Z8_6_13 + Z8_4_23 + Z8_6_23 + 2S8_3 = 2
Z9_2_13 + Z9_5_13 + Z9_7_13 + Z9_2_23 + Z9_5_23 + Z9_7_23 + 3S9_3 = 3
Z10_4_13 + Z10_7_13 + Z10_4_23 + Z10_7_23 + 2S10_3 = 2
END
```

(Constraint Set 3.27)
INTEGER $Z_{i,k,t}$, $M_{i,k,t}$, $S_{i,t}$

(Constraint Set 3.28)
GIN $Y_{k,j,t}$, $PC_{k,j,t}$, $NC_{k,j,t}$

VITA AUCTORIS

Alper Ozdemir was born in Nazilli, Turkiye on the 4th of August, 1971. He attended Bilkent University, Ankara, Turkiye from 1989 to 1993 and graduated with a B.Sc. in Industrial Engineering. He has been studying towards a M. A. Sc. in Industrial and Manufacturing Systems Engineering at the University of Windsor since September, 1993.